Universitatea
Politehnica
București

Facultatea de
Automatică și
Calculatoare

Catedra de
Calculatoare

Laborator 2

Tipuri de variabile și operatori în PL/SQL

Autori

Conf. Dr. Ing. Alexandru Boicea

As. Drd. Ing. Ciprian-Octavian Truică

Cuprins

- Tipuri de variabile și operatori în PL/SQL
- Tipuri de variabile
  - Tipuri scalare
  - Tipuri de date calendaristice
  - Tipuri corelate
  - Tipuri compuse
  - Tipuri colecție
  - Tipuri LOB (Large Objects)
- Tipuri de operatori
- Tipuri de simboluri
- Domeniul variabilelor și constantelor

Tipuri variabile și operatori PL/SQL

- PL/SQL acceptă toate tipurile de date din SQL;
- Nu sunt permise referințe anticipate, deci toate variabilele și constantele trebuie să fie declarate în secțiunea DECLARE înainte de utilizare;
- În secțiunea DECLARE trebuie declarate numele și tipul fiecărei variabile utilizate în cadrul blocului;
- Tipul specifică formatul de memorare, restricțiile și un domeniu valid de valori;
- Sintaxa declarării unei variabile este următoarea:
  - identificator [CONSTANT]{tip_de_date | identificator%TYPE | identificator%ROWTYPE} [NOT NULL] [{:= | DEFAULT} expresie_PL/SQL];

Tipuri 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.

  http://docs.oracle.com/database/121/LNPLS/datatypes.htm#LNPLS003

Tipuri scalare

- Tipurile de date ce stochează data calendaristica si 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 si NVARCHAR2;
- Tipul de date BOOLEAN stochează valori logice (true, false sau null).

Tipuri scalare

- 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;
int_an_luna INTERVAL YEAR TO MONTH :=
INTERVAL '3-2' YEAR TO MONTH; --interval de 3 ani si 2 luni
```

Tipuri scalare

- Ex. 1. Să se scrie un bloc PL/SQL care primește de la tastatură un id de angajat și va afișa un mesaj cu forma: ‘nume_angajat are un venit orar de venit_orar și face parte din departamentul denumire_departament. A lucrat în firma un număr total de zile_lucrate zile’. În caz că angajatul nu există să se trateze eroarea.

```
set serveroutput on;
declare
  numeAngajat nvarchar2(20);
  idDepartament integer;
  numeDepartament varchar(20);
  dataAngajare date;
  idAngajat number(6);
  stare boolean := true;
  venit float;
  zileLucrate number(2);
  venitOrar real;
  zileLuna constant smallint := 21;
```

Tipuri scalare

```
begin
  idAngajat := &idAngajat;
  select deptno, ename, sal + nvl(comm,0), hiredate
    into idDepartament, numeAngajat, venit, dataAngajare
    from emp where empno = idAngajat;
  select dname into numeDepartament
    from dept where deptno = idDepartament;
  venitOrar := round(venit/(zileLuna*8),2);
  zileLucrate := sysdate - dataAngajare;
  dbms_output.put_line(numeAngajat || ' are un venit orar de ' ||
      venitOrar || ' si face parte din departamentul ' ||
      numeDepartament || '.' || chr(13) || chr(10) ||
      'A lucrat in firma un numart total de ' || zileLucrate || ' zile.' );
  exception
    when no_data_found then
      stare := false;
      dbms_output.put_line('Angajatul nu se afla in baza de date!');
end;
/
```

Tipuri scalare

- Ex. 2. Să se afișeze informațiile despre ultimul angajat care a venit în firmă în anul 1982.

```
set serveroutput on;
declare
   numeAngajat varchar2(20);
   functie string(30);
   dataMax date;
   dataInceput date := '1-JAN-1982';
   dataSfarsit date := '31-DEC-1982';
```

Tipuri scalare

```
begin
  select max(hiredate) into dataMax from emp
    where hiredate between dataInceput and dataSfarsit;
  select ename, job into numeAngajat, functie
    from emp where hiredate = dataMax;
  dbms_output.put_line('Ultimul angajat venit in firma in
    anul 1982 este '||numeAngajat||' si are functia '||functie);
  exception
    when no_data_found then
      dbms_output.put_line('Nu a fost angajat nimeni in 1982.');
    when too_many_rows then
      dbms_output.put_line('Sunt mai multe angajari in ultima zi.');
end;
```

Tipuri scalare

- Ex. 3. Să se găsească care angajat are o vechime de peste 29 de ani si 6 luni in firmă.

```
set serveroutput on;
declare
   numeAngajat varchar(20);
   functie string(20);
   dataMax date;
   vechime interval year(2) to month;
```

Tipuri scalare

```
begin
   vechime := interval '29-6' year to month;
   dbms_output.put_line('Vechimea solicitata = '||vechime);
   select hiredate into dataMax from emp
     where hiredate < sysdate - vechime;
   dbms_output.put_line('Data maxima '|| dataMax);
   select ename, job into numeAngajat, functie
     from emp where hiredate = dataMax;
   dbms_output.put_line('Angajatul cu vechimea cautata este '||
     numeAngajat||' si are functia '||functie);
   exception
     when no_data_found then
       dbms_output.put_line('Nu exista angajati cu aceasta vechime.');
     when too_many_rows then
       dbms_output.put_line('Sunt mai multi angajati cu aceasta vechime.');
end;
```

Tipuri 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 corelate

- Ex. 4. Să se scrie un bloc PL/SQL care primește de la tastatură un id de angajat și afișează numele și funcția unui angajat.

```
set serveroutput on;
declare
   numeAngajat emp.ename%type;
   venit emp.sal%type;
   idDepartament emp.deptno%type;
   departament dept%rowtype;
```

Tipuri corelate

```
begin
  select deptno, ename, sal+nvl(comm,0)
    into idDepartament, numeAngajat, venit
    from emp where empno=&idAngajat;
  select * into departament from dept
    where deptno = idDepartament;
  dbms_output.put_line(numeAngajat||
    ' face parte din departamentul '||departament.dname);
  exception
    when no_data_found then
      dbms_output.put_line('Angajatul nu se gaseste in baza de date');
end;
```

Tipul înregistrare

- Tipurile scalare nu au componente interne, în timp ce tipurile compuse au componente interne care se pot manipula.
- Tipuri compuse
  - record
    - Folosit pentru a accesa liniile tabelelor sub formă de matrice
    - Asemănător cu o înregistrare a unei tabele cu excepția faptului că numele coloanelor sunt proprii fiecărei înregistrări
  - tabel
    - Este asemănător dar nu identic cu o structură tabelară

Tipul RECORD

- Declararea tipului RECORD se face conform următoarei sintaxe:

TYPE nume_tip IS RECORD

(nume_câmp1 {tip_câmp | variabilă%TYPE | nume_tabel.coloană%TYPE | nume_tabel%ROWTYPE} [[NOT NULL] {:= | DEFAULT} expresie1],

(nume_câmp2 {tip_câmp | variabilă%TYPE | nume_tabel.coloană%TYPE | nume_tabel%ROWTYPE} [[NOT NULL] {:= | DEFAULT} expresie2],

…);

Tipul RECORD

- Începând cu Oracle9i, pentru tipul RECORD sunt introduse câteva facilităţi:
  - Se poate insera (INSERT) o linie într-un tabel utilizând tipul RECORD;
  - Se poate actualiza (UPDATE) o linie într-un tabel utilizând tipul RECORD (cu sintaxa SET ROW);
  - Se poate regăsi şi returna sau şterge informaţia din clauza RETURNING a comenzilor UPDATE sau DELETE;
  - Dacă în comenzile UPDATE sau DELETE se modifică mai multe linii, atunci pot fi utilizate în sintaxa BULK COLLECT INTO, colecţii de înregistrări.

Tipul RECORD

- Ex. 5. Un exemplu de utilizare a tipului RECORD.

```
set serveroutput on;
declare
  type deptRecord is record
  (
    idDept number(2),
    numeDept varchar2(14),
    locatie varchar2(13)
  );
  recDept deptRecord;
  deptInfo dept%rowtype;
begin
  select * into deptInfo from dept where deptno=10;
  recDept := deptInfo;
  dbms_output.put_line('Denumirea departemanetului '||recDept.idDept||'
    este '||recDept.numeDept);
end;
```

Tipul RECORD

- Ex. 6. Folosindu-se tipul compus RECORD :
  - Să se creeze un record de forma tabelului dept;
  - Să se insereze un nou departament folosind o variabilă de tip record;
  - Să se selecteze înregistrarea inserată într-un record ;
  - Să se updateze înregistrarea inserată folosindu-se record și să se returneze într-un record informații;
  - Să se șteargă departamentul creat și să se returneze într-un record informații.

Tipul RECORD

```
set serveroutput on;
declare
   type myRecord is record
   (
      idDepartament dept.deptno%type,
      numeDepartament dept.dname%type,
      locatie dept.loc%type
   );
   deptRecord myRecord;
   deptRecordUpdate myRecord;
   deptRecordInsert myRecord;
```

Tipul RECORD

```
begin
   deptRecordInsert.idDepartament := 50;
   deptRecordInsert.numeDepartament := 'IT Department';
   deptRecordInsert.locatie := 'Bucuresti';
   -- insert folosind record
   insert into dept
     values(deptRecordInsert.idDepartament,
     deptRecordInsert.numeDepartament,
     deptRecordInsert.locatie);
   -- select folosind record
   select * into deptRecord from dept
     where deptno = deptRecordInsert.idDepartament;
   dbms_output.put_line('Informatii dupa insert '||
     deptRecord.idDepartament||'  '||
     deptRecord.numeDepartament||'  '||deptRecord.locatie);
```

Tipul RECORD

```
  deptRecordUpdate.idDepartament := deptRecordInsert.idDepartament;
  deptRecordUpdate.numeDepartament := 'Operatii';
  deptRecordUpdate.locatie := 'Sibiu';
  -- update frolosind record
  update dept set row = deptRecordUpdate
    where deptno = deptRecordInsert.idDepartament
    returning deptno, dname, loc into deptRecord;
  dbms_output.put_line('Informatii dupa update '||
    deptRecord.idDepartament||'  '||
    deptRecord.numeDepartament||'  '||deptRecord.locatie);
  -- stergere inregistrare
  delete from dept where deptno=deptRecordUpdate.idDepartament
    returning deptno, dname, loc into deptRecord;
  dbms_output.put_line('Informatii dupa delete '||
    deptRecord.idDepartament||'  '||
    deptRecord.numeDepartament||'  '||deptRecord.locatie);
end;
```

Tipul Colecție

- Tipuri colecție
  - Varrays
  - Index-by table
  - Nested tables

Tipul Colecție

Atribute și metode	Descriere
COUNT	numărul componentelor colecţiei
FIRST	Indicele primului element din tablou
LAST	Indicele ultimului element din tablou
EXISTS	întoarce TRUE dacă există în tablou componenta cu indexul specificat
NEXT	returnează indicele următoarei componente
PRIOR	returnează indicele componentei anterioare
DELETE	şterge una sau mai multe componente.
EXTEND	adaugă elemente la sfârşit
LIMIT	numărul maxim de elemente al unei colecţii (pentru vectori), null pentru tablouri imbricate
TRIM	şterge elementele de la sfârşitul unei colecţii

Mai multe la adresa
http://docs.oracle.com/database/121/LNPLS/composites.htm#LNPLS00508

Tipul Vector

- Vectorii (varray) sunt structuri asemănătoare vectorilor din limbajele C sau Java;
- Vectorii au o dimensiune maximă (constantă) stabilită la declarare;
- Se utilizează pentru modelarea relaţiilor one-to-many, atunci când numărul maxim de elemente din partea „many“ este cunoscut şi ordinea elementelor este importantă;
- Fiecare element are un index, a cărui limită inferioară este 1;
- Tipul de date vector este declarat utilizând sintaxa:

  TYPE nume_tip IS {VARRAY | VARYING ARRAY} (lungime_maximă) OF tip_elemente [NOT NULL]

Tipul Vector

- Ex. 7. Exemplul folosire varray

```
set serveroutput on;
declare
  type secventa is varray(5) of varchar2(10);
  v_sec secventa := secventa ('alb', 'negru', 'rosu', 'verde');
begin
  dbms_output.put_line(v_sec(1)||' '||v_sec(2)||' '||v_sec(3)||' '||v_sec(4));
  v_sec(4) := 'rosu';
  dbms_output.put_line(v_sec(1)||' '||v_sec(2)||' '||v_sec(3)||' '||v_sec(4));
  --desi s-au declarat 5 elemente pentru vector acest print va da eroare
  --dbms_output.put_line(v_sec(5));
  v_sec.extend; -- adauga un element null
  dbms_output.put_line(v_sec(1)||' '||v_sec(2)||' '||v_sec(3)||' '||v_sec(4)
    ||' '||v_sec(5));
  v_sec(5) := 'albastru';
  dbms_output.put_line(v_sec(1)||' '||v_sec(2)||' '||v_sec(3)||' '||v_sec(4)
    ||' '||v_sec(5));
  --extinderea la 6 elemente va genera o eroarea
  --v_sec.extend;
end;
```

Tipul Index-by Table

- Un tabloul indexat în PL/SQL are două componente:
  - coloană ce cuprinde cheia primară pentru a avea acces la liniile tabloului;
  - o coloană care include valoarea efectivă a elementelor tabloului.
- Declararea tipului TABLE se face cu următoarea sintaxă:

```
TYPE nume_tip IS TABLE OF
{tip_coloană | variabilă%TYPE |
nume_tabel.coloană%TYPE [NOT NULL] |
nume_tabel%ROWTYPE}
INDEX BY tip_indexare;
```

Tipul Index-by Table

- Observații:
  - Elementele unui tablou indexat nu sunt într-o ordine particulară şi pot fi inserate cu chei arbitrare;
  - Deoarece nu există constrângeri de dimensiune, dimensiunea tabloului se modifică dinamic;
  - Tabloul indexat nu poate fi iniţializat în declararea sa;
  - Un tablou indexat neiniţializat este vid (nu conţine nici valori, nici chei);
  - Un element al tabloului este nedefinit atâta timp cât nu are atribuită o valoare efectivă;
  - Dacă se face referire la o linie care nu există, atunci se produce excepţia NO_DATA_FOUND.

Tipul Index-by Table

- Ex. 8. Exemplu de folosire Index-by Table

```
set serveroutput on;
declare
  type tabIndex is table of dept%rowtype index by binary_integer;
  myTabIndex tabIndex;
  rowDept dept%rowtype;
begin
  select * into rowDept from dept where deptno = 10;
  myTabIndex(1) := rowDept;
  select * into myTabIndex(2) from dept where deptno = 20;
  dbms_output.put_line(rpad('Id', 20)||rpad('Nume',20)||lpad('Locatie',20));
  dbms_output.put_line(rpad(myTabIndex(1).deptno, 20)||
    rpad(myTabIndex(1).dname,20)||lpad(myTabIndex(1).loc,20));
  dbms_output.put_line(rpad(myTabIndex(2).deptno, 20)||
    rpad(myTabIndex(2).dname,20)||lpad(myTabIndex(2).loc,20));
end;
```

Tipul Nested Table

- Tablourile imbricate (nested table) sunt tablouri a căror dimensiune nu este stabilită.
  - folosesc drept indici numere consecutive;
  - sunt asemenea unor tabele cu o singură coloană;
  - nu au dimensiune limitată, ele cresc dinamic;
  - iniţial, un tablou imbricat este dens (are elementele pe poziţii consecutive) dar pot apărea spaţii goale prin ştergere;
  - metoda NEXT ne permite să ajungem la următorul element;
  - pentru a insera un element nou, tabloul trebuie extins cu metoda EXTEND(nr_comp).

Tipul Nested Table

- Un tablou imbricat este o mulţime neordonată de elemente de acelaşi tip. Valorile de acest tip:
  - pot fi stocate în baza de date;
  - pot fi prelucrate direct în instrucţiuni SQL;
  - au excepţii predefinite proprii.
- Comanda de declarare a tipului de date tablou imbricat are sintaxa:

  TYPE nume_tip IS TABLE OF tip_elemente [NOT NULL];

Tipul Nested Table

- Pentru adăugarea de linii intr-un tablou imbricat, acesta trebuie sa fie inițializat cu ajutorul constructorului.
  - PL/SQL apelează un constructor numai în mod explicit;
  - Tabelele indexate nu au constructori;
  - Constructorul primeşte ca argumente o listă de valori numerotate în ordine, de la 1 la numărul de valori date ca parametrii constructorului;
  - Dimensiunea iniţială a colecţiei este egală cu numărul de argumente date în constructor, când aceasta este iniţializată;
  - Pentru vectori nu poate fi depăşită dimensiunea maximă precizată la declarare.
  - Atunci când constructorul este fără argumente va crea o colecție fără niciun element (vida), dar care are valoarea *not null*.

Tipul Nested Table

- Ex. 9. Exemplu de folosire al unui tabel imbricat.

```
set serveroutput on;
declare
  type tabImbricat is table of varchar2(30);
  curs tabImbricat;
begin
  curs := tabImbricat('Baze de date', 'Proiectare Baze de Date');
  dbms_output.put_line('Cursul echivalent pentru BD2 este '||curs(2));
end;
```

Tipul colecție

- Tipurile colecție pot fi definite direct în baza de date

Collection Type	Number of Elements	Subscript Type	Dense or Sparse	Where Created	Can Be Object Type Attribute
Associative array (or index-by table)	Unbounded	String or integer	Either	Only in PL/SQL block	No
Nested table	Unbounded	Integer	Starts dense, can become sparse	Either in PL/SQL block or at schema level	Yes
Variable-size array (varray)	Bounded	Integer	Always dense	Either in PL/SQL block or at schema level	Yes

Tipul colecție

- Ex. 10. Definirea tipurilor direct în baza de date

```
create or replace type myVarray is varray(30) of varchar(30);
/
create or replace type myTabNested as table of varchar2(30);
/
-- folosind vector
create type listaProiecte as varray(20) of varchar(20);
/
create table proiecte(
  idDept number(2),
  denumire varchar2(20),
  buget number(11,2),
  proiect listaProiecte
);
/
--folosind nested table
create type listaProiecte_nt as table of varchar2(30);
/
create table proiecte_nt(
  idDept number(2),
  denumire varchar2(20),
  buget number(11,2),
  proiect listaProiecte_nt
)
nested table proiect store as proiect_tab;
```

Tipul colecție

```
set serveroutput on;
declare
  proiect listaProiecte;
begin
  proiect := listaProiecte('E-commerce','Carduri bancare');
  insert into proiecte values(20, 'Proiectare', 165580, proiect);
  dbms_output.put_line('Proiectul bancar se numeste '||proiect(2));
end;
```

Tipul colecție

- Pentru a insera mai multe valori într-o variabilă de tip colecție se poate folosi BULK COLLECT INTO;
- În următoarele exemple este folosită instrucțiunea FOR despre care vom vorbi în laboratoarele următoare.

Tipul colecție

- Ex. 11. Folosirea BULK COLLECT INTO cu varray.

```
set serveroutput on;
declare
  type vector is varray(4) of dept.deptno%type;
  myVector vector;
  contor integer;
 begin
  dbms_output.put_line('folosire bulk collect into varray');
  select deptno bulk collect into myVector from dept;
  for contor in myVector.first .. myVector.last
  loop
    dbms_output.put_line('myVector('||contor||')='||myVector(contor));
  end loop;
end;
```

Tipul colecție

- Ex. 12. Folosirea BULK COLLECT INTO cu index-by table.

```
set serveroutput on;
declare
  type tableIndexat is table of dept%rowtype index by binary_integer;
  contor integer;
  tblIndexat tableIndexat;
begin
  dbms_output.put_line('folosire bulk collect into index-by table');
  select * bulk collect into tblIndexat from dept;
  for contor in tblIndexat.first .. tblIndexat.last
  loop
    dbms_output.put_line('contor='||contor
      ||' tblIndexat.deptno='||tblIndexat(contor).deptno
      ||' tblIndexat.dname='||tblIndexat(contor).dname
      ||' tblIndexat.loc='||tblIndexat(contor).loc);
  end loop;
end;
```

Tipul colecție

- Ex. 13. Folosirea BULK COLLECT INTO cu nested table.

```
set serveroutput on;
declare
  type tableNested is table of emp%rowtype;
  contor integer;
  tblNested tableNested;
begin
  dbms_output.put_line('folosire bulk collect into nested table');
  select * bulk collect into tblNested from emp where deptno=10;
  for contor in tblNested.first .. tblNested.last
  loop
    dbms_output.put_line('contor='||contor
      ||' tblNested.empno='||tblNested(contor).empno
      ||' tblNested.ename='||tblNested(contor).ename
      ||' tblNested.sal='||tblNested(contor).sal
      ||' tblNested.comm='||tblNested(contor).comm
      ||' tblNested.hiredate='||tblNested(contor).hiredate
      ||' tblNested.mgr='||tblNested(contor).mgr
      ||' tblNested.deptno='||tblNested(contor).deptno);
  end loop;
end;
```

Tipul colecție

- Pentru a crea un *table* care sa aibă coloane definite de dezvoltatori se va folosi un *table*(*index-by* sau *nested*) care să fie de tipul RECORD.

Tipul colecție

- Ex. 14. Crearea unui tabel cu coloanele definite de utilizator.

```
set serveroutput on;
declare
  type myRecord is record(
    numeAngajat emp.ename%type,
    numeDepartament dept.dname%type
  );
  type myTabel is table of myRecord;
  tbl myTabel;
  contor integer;
begin
  select e.ename, d.dname bulk collect into tbl
    from emp e natural join dept d;
  for contor in tbl.first .. tbl.last
  loop
    dbms_output.put_line(tbl(contor).numeAngajat||' lucreaza in departamentul '
      ||tbl(contor).numeDepartament);
  end loop;
end;
```

Tipuri compuse și colecții

- Mai există și tipul OBJECT care nu este prezentat în laborator, găsiți mai multe detalii despre acest tip la adresa:

https://docs.oracle.com/database/121/ADOBJ/adobjint.htm#ADOBJ001

- Mai multe detalii la adresa:

http://docs.oracle.com/database/121/LNPLS/composites.htm#LNPLS005

Tipul LOB

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

Tipul LOB

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.

Observații

- Deoarece PL/SQL este o extensie pentru SQL, regulile sintactice de bază aplicabile în SQL sunt valabile și în PL/SQL:
  - Instrucțiunile pot avea mai multe linii cu excepția cuvintelor cheie;
  - Unitățile lexicale (identificatori, nume de operatori) pot fi separate de unul sau mai multe spații, sau alt delimitator care nu poate face parte din unități lexicale;
  - Cuvintele rezervate nu pot fi folosite ca identificatori decât între ghilimele duble;
  - Identificatorii trebuie să înceapă cu o literă și pot conține cel mult 30 caractere
  - Șirurile trebuie să fie marcate cu ghilimele simple;
  - Numerele pot fi reprezentate prin valoarea lor sau științifică;
  - Comentariile pe mai multe linii se delimitează cu /* ..... */
  - Comentariile pe o singură linie se marchează cu două linii orizontale --

Tipuri de operatori

Operator	Descriere
+	Adunare
-	Scădere
*	Multiplicare
/	Împărțire
**	Exponențial
=	Egalitate
>	Mai mare
<	Mai mic
<>	Diferit de
!=	Diferit de (Unix)
^=	Diferit de (IBM)
<=	Mai mic sau egal
>=	Mai mare sau egal
%	Coincidență parțială

Tipuri de operatori

Operator	Descriere
NOT BETWEEN ... AND ...	Nu se află între două valori date
NOT IN (list)	Nu se află într-o listă de valori
NOT LIKE	Diferit de șir
IS NOT NULL	Diferit de valoare nulă
NOT numecoloana =	Diferit de
NOT numecoloana {>|<|<=|>=}	Mai mic sau egal | Mai mare sau egal | Mai mare | Mai mic
BETWEEN ... AND ...	Între două valori
IN (list)	Cuprins într-o listă
LIKE	Comparație cu șir de caractere
IS NULL	Este valoare nulă

Tipuri de simboluri

- Ordinea în care sunt realizate operațiile poate fi controlată de paranteze.
- Ordinea naturală a operațiilor în cadrul unei expresii este următoarea:

Ordine	Operație	Descriere
1	** , NOT	exponențial, negație
2	+, -	identitate, negație
3	*, /	înmulțire, împărțire
4	+, -, ||	adunare, scădere, concatenare
5	=, !=, <, >, <=,>=,IS NULL, LIKE, BETWEEN, IN	comparație
6	AND	conjuncție
7	OR	incluziune

Tipuri de simboluri

- Tipuri de simboluri simple:

Simbol	Descriere
( )	Delimitator de listă sau expresii
;	Delimitator de instrucțiuni
,	Separator de obiect
.	Selector de componentă
@	Delimitator de acces la baza de date
‘	Delimitator de șir
:	Delimitator de server
$	Format de afișare în USD
&	Variabilă substituită

Tipuri de simboluri

- Tipuri de simboluri compuse:

Simbol	Descriere
:=	Asignare
=>	Asociere
..	Rang
||	Concatenare
<< >>	Etichetă
--	Comentariu pe o singură linie
/* */	Comentatiu pe mai multe linii

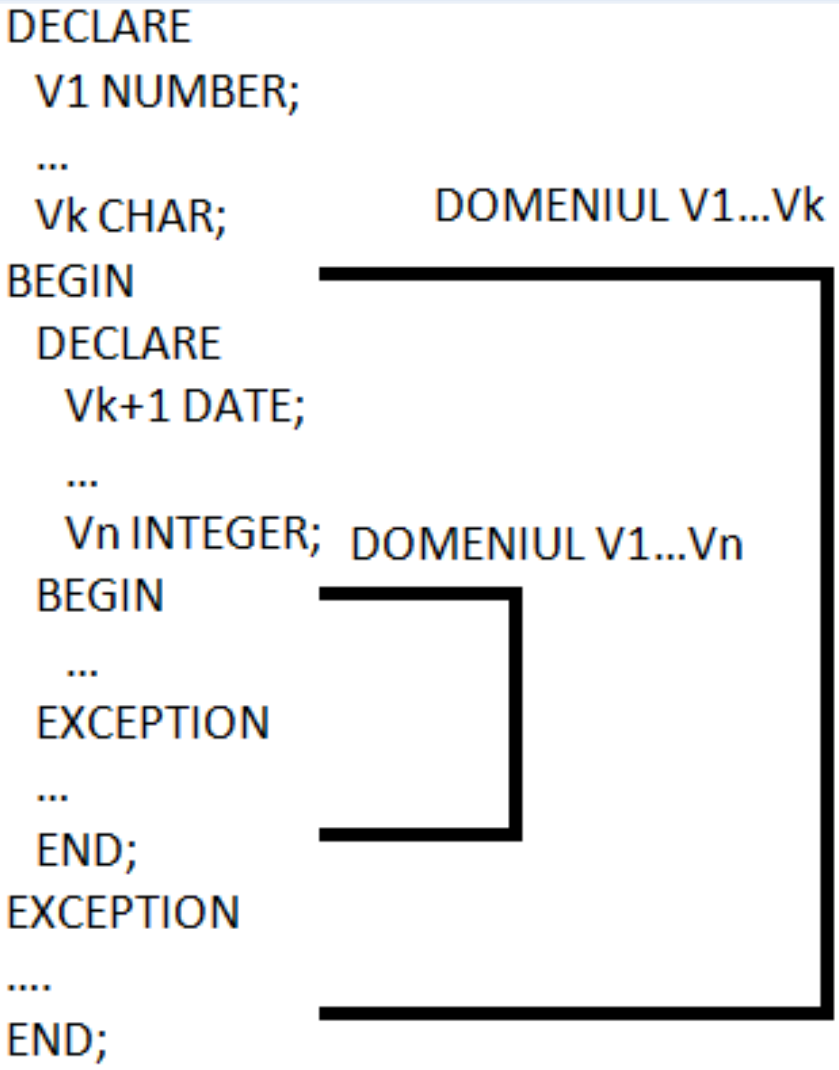
Obs. Nu sunt permise spații între cele două caractere ale simbolurilor compuse.

Domeniul variabilelor și constantelor

- Domeniul de valabilitate al variabilelor reprezintă zona unui program în care acestea pot fi folosite;
- Variabilele și constantele pot fi declarate în cadrul blocului în care au fost definite sau în orice sub-bloc inclus în acesta;
- Variabilele declarate într-un sub-bloc pot fi folosite doar în acesta sau în sub-blocurile acestuia;
- Variabilele se pot transmite de la un bloc la sub-bloc, dar nu și invers;
- Aceste constrângeri se aplică tuturor obiectelor declarate, inscusiv cursoare, constante și excepții definite de utilizator.

Domeniul variabilelor și constantelor

- Se observă că în sub-bloc se pot folosi atât variabilele definite local cât și cele din blocul superior;
- Dacă într-un sub-bloc se definește o variabilă cu același nume ca una dintr-un bloc superior atunci are prioritate cea locală, dezactivându-se variabila definită în blocul superior;
- Se recomandă să se evite astfel de situații pentru a nu crea confuzii.



Domeniul variabilelor și constantelor

- PL/SQL permite o varietate de tipuri de date ce pot fi folosite pentru declararea de variabile și constante;
- Variabilele pot avea, opțional, atribuite valori la declarare și pot să-și schimbe valoarea prin atribuiri ulterioare în cadrul blocului;
- Constantele sunt identificatori care păstrează o valoare fixă ce trebuie atribuită în momentul declarării;
- Instructiunile PL/SQL nu sunt *case sensitive*;
- Setul de caractere este format din literele mari și mici, cifrele de la 0-9, spații, caractere de tabulare, carriage return și simboluri speciale.