Lucrarea 8.
PL/SQL. Blocuri anonime. Variabile.

PL/SQL (Procedural Language SQL)

Extensie a limbajului SQL care permite:

- Comenzi SQL executate de serverul SQL Oracle
- Comenzi procedurale care permit programe ramificate si ciclice
- Declarare de variabile
- Utilizare variabile de legatura cu SQL+ (:var)
- Structuri de date tip RECORD si TABLE
- Utilizare functii si proceduri PL/SQL standard (sute)
- Definirea de proceduri si functii stocate pe server
- Definirea de Package-uri ca seturi de functii si proceduri
- Definire de CURSOR pentru accesul la inregistrari
- Definire TRIGGER pentru declansarea unor proceduri la comenzi DML
- Definire clase de obiecte cu proprietati si metode proprii

Programele PL/SQL se editeaza in fisiere .sql prin comanda:
EDIT pr1 - editeaza un script pr1.sql
START pr1 - lanseaza in executie progamul
In programele PL/SQL se admit comentarii pe linia de program precedate de - -
si pe mai multe linii, sau in comanda delimitate prin /* text comentariu */
Fiecare linie de program se termina cu **;**
Programul cuprinde o zona de declaratii precedata de **DECLARE** si o zona procedurala precedata de **BEGIN.**
Lansarea in executie a unui bloc de program PL/SQL se face prin / **.**
Conversiile din numeric in Character se fac si direct.
Instructiunea de atribuire este : =

Structura unui bloc PL/SQL

```
SET     VERIFY OFF                  - suprima afisare valori vechi la substitutie
-- comanda pentru validare apel functii PL/SQL si lungime buffer
SET serveroutput ON size 20000 format WORD_WRAPPED
DECLARE
-- Declaratie variabile si constante
PI  CONSTANT NUMBER(7.2) :=3.14;      - declarare constanta
V_Nume      Varchar2(20);         -- definire  variabila character
V_salar     Number(9);            -- definire variabila numerica
V_impozit   Number(7) :=0;        -- definire variabila cu valoare initiala
V_azi       Date;
BEGIN
V_azi:= SYSDATE;
--  Un Select va returna valori dintr-un singur rand memorate in variabile declarate
SELECT Ename, Sal INTO v_Nume,V_Salar FROM EMP WHERE Empno='7902';
V_impozit:= V_Salar*0.25;   -- calcul impozit
/*  Afisare rezulte prin functia de afisare o linie */
```

```
Dbms_output.put_line( 'Numele:  ' || V_Nume || '  Salar: ' || TO_Char(V_Salar)
        || '  Impozit:  ' || To_Char(V_impozit));
/* Selectie cu dialog si substitutie – se cere codul salariatului - cods*/
SELECT Ename, Sal INTO V_Nume,V_Salar FROM EMP WHERE Empno=&cods;
Dbms_output.put_line( 'Numele:  ' || V_Nume || '  Salar: ' || TO_Char(V_Salar)
        || '  Impozit:  ' || To_Char(V_impozit));
-- Afisare numar cu conversie directa
Dbms_output.put_line(' Valoarea lui PI=' || PI);
-- Afisare data cu conversie directa
Dbms_output.put_line( ' Astazi: ' || SYSDATE);
Dbms_output.put_line( ' Astazi: ' || V_Azi);
-- Afisare data cu conversie explicita
Dbms_output.put_line( ' Astazi: ' ||
 To_CHAR(V_Azi,'dd-month-yyyy-year-day-hh24-mi-ss'));

END;            -- sfarsit bloc PL/SQL
*   lansare in executie bloc PL/SQL
/
```

Variabile de legatura

Variabilele de legatura se utilizeaza pentru transmiterea valorii unor variabile din PL/SQL spre SQL+. Ele se declara in SQL+ si se folsesc in PL/SQL precedate de doua puncte (:nume, :salary) ca in exemplul urmator.
Aceste variabile se pot afisa cu PRINT in SQL+.

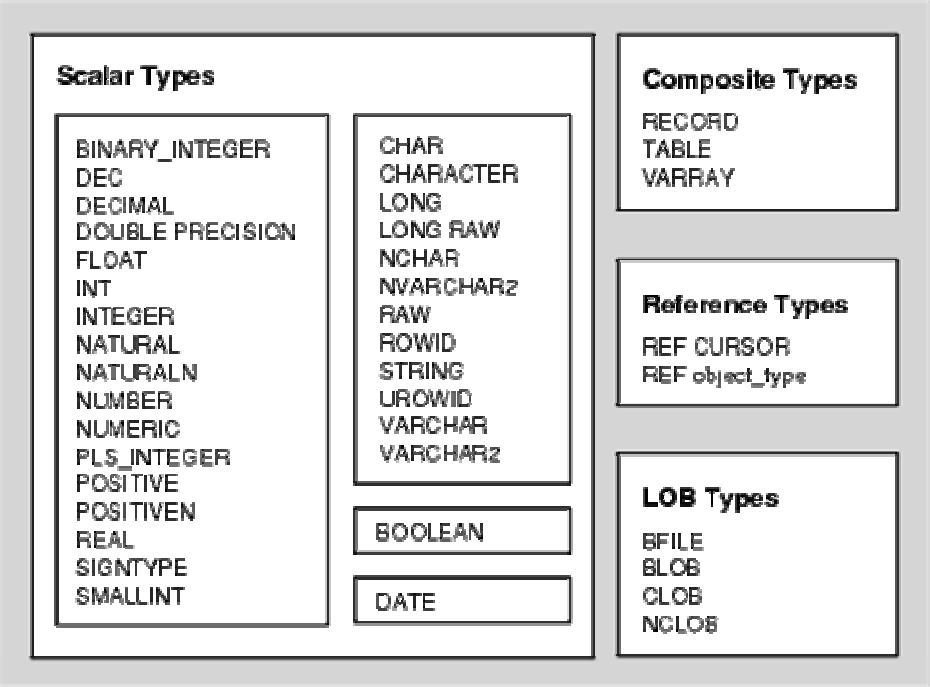
```
/* Program de calcul salar global
  se da nume salariat sau conditia
  se utilizeaza variabile de legatura cu PL/SQL -Nume si Salary*/

Variable salary number
Variable Nume Char(20)
*    dialog in SQL+
* accept salary prompt ' salariul:  '
* -- accept nume prompt 'Nume salariat:  '
* -- accept  ecod prompt 'Cod salariat:  '
declare       -- declarare variabile PL/SQL
a1 char(15);
a2 number(7);

begin               -- Bloc de calcul

-- Selectie si calcul .Valorile memorate in variabile PL/SQL
Select ename, sal*12+nvl(comm,0) into a1,a2
 from emp  where empno=7369;
/*-- ename like (S%)
-- epno=&ecod
-- &conditie  */
:salary:=a2;       -- atribuire valoare pentru variabila de legatura salary
:nume :=a1;
end;
/
* Afisare rezultat in SQL+
print Nume salary
```

Datatypes

Every constant and variable has a *datatype*, which specifies a storage format, constraints, and valid range of values. PL/SQL provides a variety of predefined datatypes. A *scalar* type has no internal components. A *composite* type has internal components that can be manipulated individually. A *reference* type holds values, called *pointers*, that designate other program items. A `LOB` type holds values, called lob locators, that specify the location of large objects (graphic images for example) stored out-of-line.

Figure 2-1 shows the predefined datatypes available for your use. The scalar types fall into four families, which store number, character, Boolean, and date/time data, respectively.

Figure 2-1 Built-in Datatypes



Built-In Functions

PL/SQL provides many powerful functions to help you manipulate data. These built-in functions fall into the following categories:

- error reporting
- number
- character
- datatype conversion
- date
- object reference
- miscellaneous

Table 2-4 shows the functions in each category. For descriptions of the error-reporting functions, see Chapter 11. For descriptions of the other functions, see *Oracle8i SQL Reference*.

Except for the error-reporting functions SQLCODE and SQLERRM, you can use all the functions in SQL statements. Also, except for the object-reference functions DEREF, REF, and VALUE and the miscellaneous functions DECODE, DUMP, and VSIZE, you can use all the functions in procedural statements.

Although the SQL aggregate functions AVG, COUNT, GROUPING, MIN, MAX, SUM, STDDEV, and VARIANCE are not built into PL/SQL, you can use them in SQL statements (but not in procedural statements).

Table 2-4 Built-in Functions

Error	Number	Character	Conversion	Date	Obj Ref	Misc
SQLCODE	ABS	ASCII	CHARTOROWID	ADD_MONTHS	DEREF	BFILENAME
SQLERRM	ACOS	CHR	CONVERT	LAST_DAY	REF	DECODE
	ASIN	CONCAT	HEXTORAW	MONTHS_BETWEEN	VALUE	DUMP
	ATAN	INITCAP	RAWTOHEX	NEW_TIME		EMPTY_BLOB
	ATAN2	INSTR	ROWIDTOCHAR	NEXT_DAY		EMPTY_CLOB
	CEIL	INSTRB	TO_CHAR	ROUND		GREATEST
	COS	LENGTH	TO_DATE	SYSDATE		LEAST
	COSH	LENGTHB	TO_MULTI_BYTE	TRUNC		NLS_CHARSET_DECL_LEN
	EXP	LOWER	TO_NUMBER			NLS_CHARSET_ID
	FLOOR	LPAD	TO_SINGLE_BYTE			NLS_CHARSET_NAME
	LN	LTRIM				NVL
	LOG	NLS_INITCAP				SYS_CONTEXT
	MOD	NLS_LOWER				SYS_GUID
	POWER	NLS_UPPER				UID
	ROUND	NLSSORT				USER
	SIGN	REPLACE				USERENV
	SIN	RPAD				VSIZE
	SINH	RTRIM				
	SQRT	SOUNDEX				
	TAN	SUBSTR				
	TANH	SUBSTRB				
	TRUNC	TRANSLATE				
		TRIM				
		UPPER				

*** Utilizare functii agregat in PL/SQL**

* Test1 * Variabile externe

```
set serveroutput on size 20000
set pagesize 50
set linesize 120
set verify off
variable var1 char(5)
variable v2 number
```

* Urmeaza programul PL/SQL

```
declare
v1 char(5):='&v1';
--v2 number;
begin
Select avg(sal) into :v2 from emp;
dbms_output.put_line('Medie salarii este V2= ' ||:v2 );
:var1:=v1;
dbms_output.put_line(' var1 este:'||:var1||'V1= ' ||v1 );
end;
/
defi var1
print var1
print v2
```