

Fisa individuala Practica SGBD

Constantinescu George-Gabriel
grupa E3

24 Februarie 2021

1 Blocuri anonime

PL/SQL este un limbaj structurat in blocuri, astfel bucatile de cod sunt structurate in mai multe blocuri de date independente una de cealalta.

Un bloc PL/SQL este compus din mai multe sectiuni:

- Sectiunea declarativa
- Sectiunea executabila
- Sectiunea de exceptii

In PL/SQL, un bloc are asociat un nume. De exemplu: functiile si procedurile sunt blocuri care au asociat un nume. Un bloc de acest tip este intotdeauna stocate de catre SGBD si pot fi reutilizate.

Un bloc care nu are un nume asociat se numeste **Bloc Anonim**. Astfel, blocurile anonime, spre deosebire de cele care au asociat un nume nu vor fi retinute de catre SGBD. Astfel, putem deduce un avantaj al blocurilor anonime: se pot folosi pentru testare.

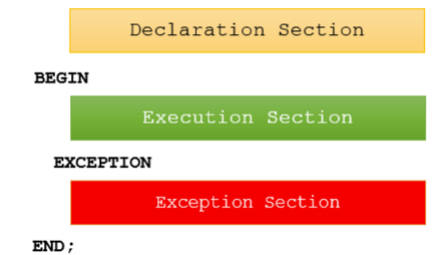


Figure 1: Structura unui bloc

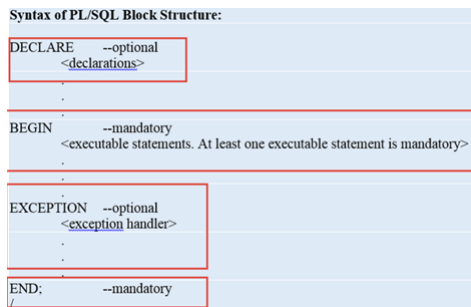


Figure 2: Sintaxa unui bloc PL/SQL

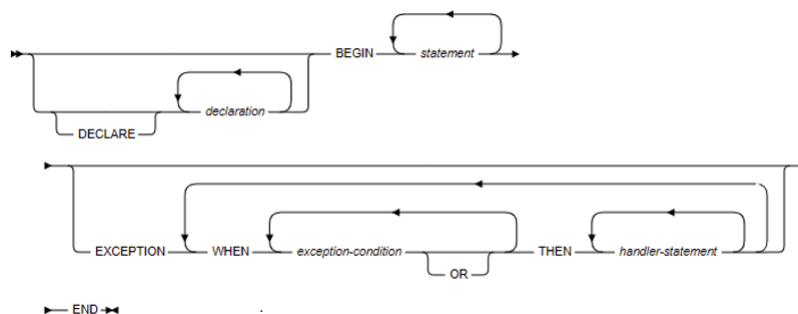
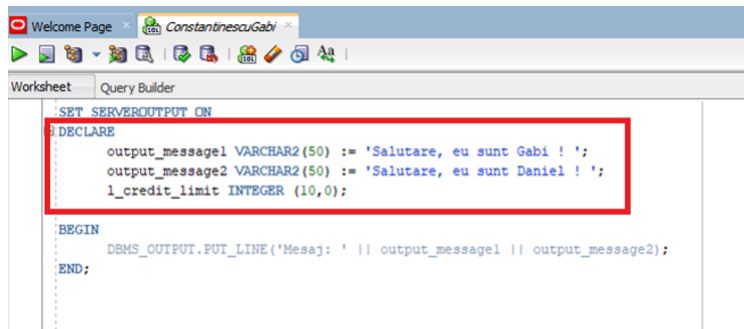


Figure 3: Sintaxa unui bloc PL/SQL

Structura blocurilor anonime este formata din:

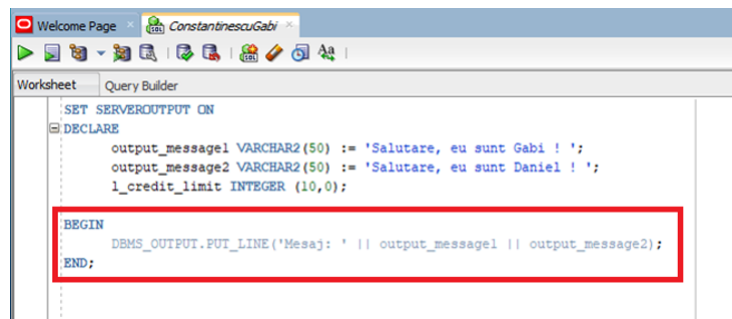
- **Sectiunea declarativa:** se pot declara variabile, se poate alocă memorie pentru cursori si se pot define tipuri de date. Aici se mai pot declare deasemenea si constante sau exceptii. Putem identifica aceasta sectiune destul de usor: sectiunea declarativa incepe prin cuvantul-cheie **DECLARE** .
- **Sectiunea executabila:** Aceasta sectiunea incepe cu cuvantul cheie **BEGIN** si se incheie cu cuvantul cheie **END** . Aceasta sectiune contine instructiuni SQL sau PL/SQL , necesare pentru manipularea informatiilor din baza de date.
- **Sectiunea de exceptii:** Aceasta sectiune este o parte din sectiunea executabila. Intotdeauna aceasta incepe la finalul sectiunii executabile si este caracterizata prin cuvantul cheie **EXCEPTION**, care marcheaza inceputul acestei sectiuni. Aceasta sectiune are un rol foarte important fata de sectiunea executabila deoarece in momentul in care se produc erori la nivelul sectiunii executabile, in aceasta sectiunea pot fi definite secvente de cod ce trateaza exceptiile care se incalca in partea de cod executabil.



```
SET SERVEROUTPUT ON
DECLARE
    output_message1 VARCHAR2(50) := 'Salutare, eu sunt Gabi !';
    output_message2 VARCHAR2(50) := 'Salutare, eu sunt Daniel !';
    l_credit_limit INTEGER (10,0);

BEGIN
    DBMS_OUTPUT.PUT_LINE('Mesaj: ' || output_message1 || output_message2);
END;
```

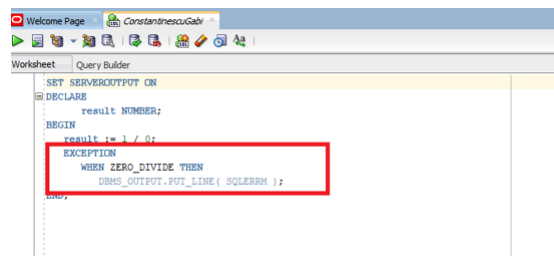
Figure 4: Sectiunea declarativa



```
SET SERVEROUTPUT ON
DECLARE
    output_message1 VARCHAR2(50) := 'Salutare, eu sunt Gabi !';
    output_message2 VARCHAR2(50) := 'Salutare, eu sunt Daniel !';
    l_credit_limit INTEGER (10,0);

BEGIN
    DBMS_OUTPUT.PUT_LINE('Mesaj: ' || output_message1 || output_message2);
END;
```

Figure 5: Sectiunea executabila



```
SET SERVEROUTPUT ON
DECLARE
    result NUMBER;
BEGIN
    result := 1 / 0;
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('SQLERRM');
END;
```

Figure 6: Sectiunea de exceptii

2 Variabile

În PL / SQL, o variabilă reprezintă o locație de stocare care are asignată un nume și stochează o valoare a unui anumit tip de date. Valoarea variabilei se modifică prin intermediul instrucțiunilor din program. Înainte de a utiliza o variabilă, trebuie să o declarați în secțiunea de declarație a unui bloc.

```
variable_name datatype [NOT NULL] [:= initial_value];
```

Figure 7: Forma unei variabile

- Variablename — > numele variabilei respective
- Datatype — > tipul variabilei respective
- NOT NULL — > dacă poate sau nu reține valoarea NULL
- Initialvalue — > valoarea cu care initializăm variabile

2.1 Declararea unei variabile

În primul rând, trebuie să specificăm numele variabilei. Acest trebuie să fie cât mai descriptiv posibil. Exemple: l_total_cheltuieli, l_total_venituri, l_sales_profit.

În al doilea rând, trebuie să un tip de date adecvat variabilei în funcție de tipul de valoare pe care dorim să o stocăm. De exemplu: number, character, Boolean, and datetime.

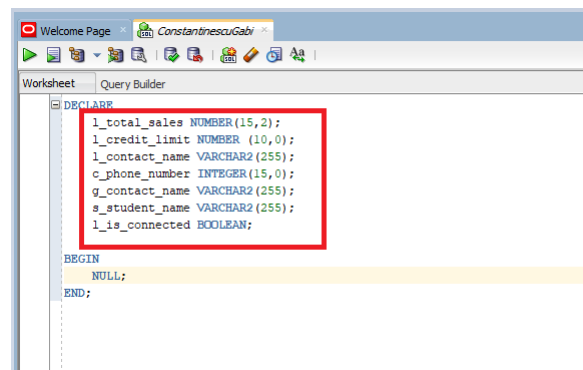


Figure 8: Exemple

Se mai pot folosi si alte tipuri de date cum ar fi:

Data Type	Synonyms
NUMBER	DEC, DECIMAL, DOUBLE PRECISION, FLOAT, INTEGER, INT, NUMERIC, REAL, SMALLINT
CHAR	CHARACTER, STRING
VARCHAR2	VARCHAR

Observatie1: VARCHAR2 consuma mai mult spatiu decat lungimea cuvantului cu care este initializat. Insa, tipu de date CHAR va ocupa exact atat cu cat a fost initializat.

Observatie2: Tipul de date VARCHAR2 nu poate sesiza diferenta intre un string gol si NULL. Insa, tipul de date VARCHAR poate face acest lucru.

Exemplu:

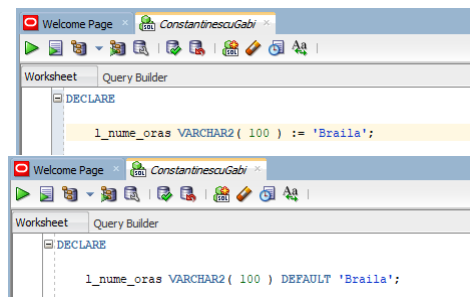


Figure 9: In acest caz, cele 2 blocuri sunt echivalente.

In cazul in care declararea contine si NOT NULL, se impune in locul in care este declarata variabila respectiva sa se initializeze.

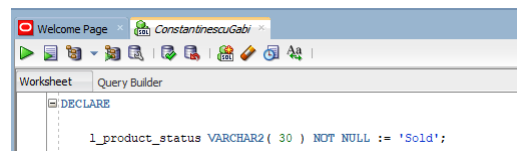


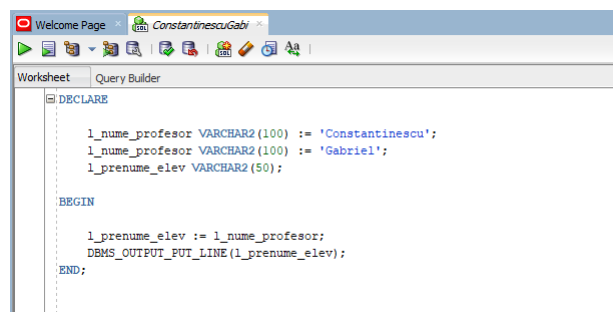
Figure 10: Exemplu NOT NULL.

In caz contrar, se va afisa urmatoarea eroare:

Error report - ORA-06550: line 3, column 22: PLS-00218: a variable declared NOT NULL must have an initialization assignment	ORA-06502: PL/SQL: numeric or value error
---	---

Figure 11: Exemple de eroare.

Initializarea variabilelor in blocuri se poate face ori in blocul de date al declaratiilor, ori in blocul de date al executiei.

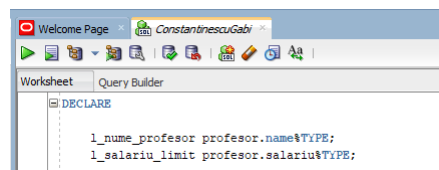


```
DECLARE
    l_nume_profesor VARCHAR2(100) := 'Constantinescu';
    l_nume_profesor VARCHAR2(100) := 'Gabriel';
    l_prenume_elev VARCHAR2(50);

BEGIN
    l_prenume_elev := l_nume_profesor;
    DBMS_OUTPUT.PUT_LINE(l_prenume_elev);
END;
```

Figure 12: Exemplu cu initializari

In cazul in care vrem sa selectam o valoarea dintr-o coloana a unui tabel, putem folosi declarare cu %TYPE.



```
DECLARE
    l_nume_profesor profesor.name%TYPE;
    l_salariu_limit profesor.salariu%TYPE;
```

Figure 13: Exemplu cu %TYPE

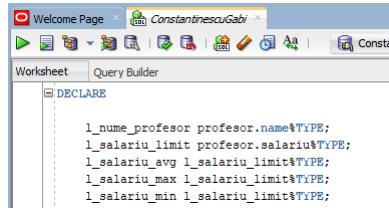


Figure 14: Exemplu cu %TYPE, se foloseste declararea unei variabile initializate

2.2 Domeniu de vizibilitate

Conform domeniului de vizibilitate, o variabila poate fi:

- Globala: variabila este declarata intr-un block si accesata in acelasi block.
- Locala: variabila este declarata intr-un block dar accesata in toate celelalte blocuri.

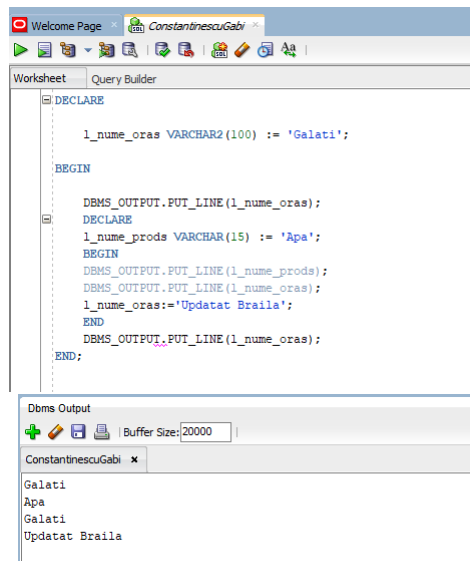


Figure 15: Exemplu cu mai multe variabile globale si locale.

3 Constante

O variabila care este declarata constant nu i se poate schimba ulterior valoarea pe parcursul executiei.

```
l_pret_final := l_pret_final + l_pret_final * 0.25;  
  
l_pret_final := l_pret_final + l_pret_final * bonus.Variation;
```

Figure 16: Putem observa ca utilizand constante codul este mai usor de inteles.

3.1 Declararea

```
constant_name CONSTANT datatype [NOT NULL] := expression
```

Figure 17: Declararea unei variabile CONSTANT

- ConstantName – > numele variabilei de tip constant
- Datatype – > tipul variabilei respective
- NOT NULL – > daca poate sau nu retine valoarea NULL
- Expression – > valoarea cu care initializam constanta

Mai jos se pot vedea mai multe exemple cu constante:

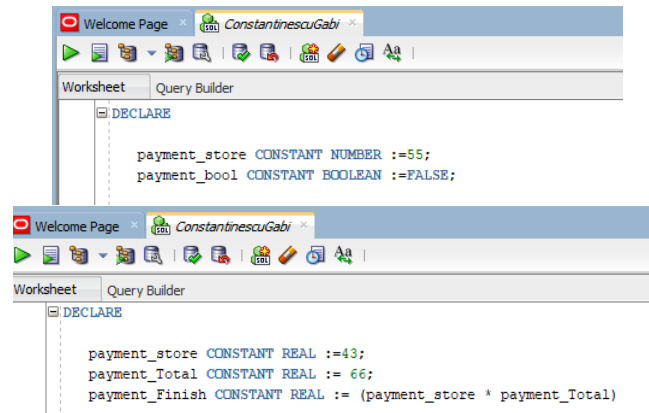


Figure 18: Exemple cu variabile constante.

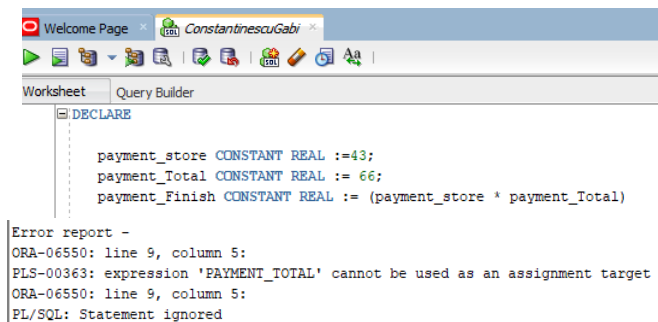


Figure 19: Aici este un contraexemplu prin care putem vedea ce eroare se genereaza in cazul in care schimbam valoarea unei variabile de tip constant.

4 Tipuri de date

Fiecare constantă, variabilă și parametru are un tip de date (numit și data type) care determină formatul de stocare, constrângerile, valorile suportate dar și operațiile care pot fi efectuate în PL/SQL.

Data Type Category	Data Description
Scalar	Single values with no internal components.
Composite	Data items that have internal components that can be accessed individually. Explained in Chapter 5, "Using PL/SQL Collections and Records."
Reference	Pointers to other data items. Explained in Using Cursor Variables (REF CURSORS) .
Large Object (LOB)	Pointers to large objects that are stored separately from other data items, such as text, graphic images, video clips, and sound waveforms.

Category	Data Description
Numeric	Numeric values, on which you can perform arithmetic operations.
Character	Alphanumeric values that represent single characters or strings of characters, which you can manipulate.
BOOLEAN	Logical values, on which you can perform logical operations.
Datetime	Dates and times, which you can manipulate.
Interval	Time intervals, which you can manipulate.

Figure 20: Tabele Tipuri de Variabile.

- Scalar – > componentele constante care nu detin alte componente.
- Composite – > componentele au la randul sau alte componente care pot fi accesate individual.
- Reference – > pointeri catre alte tipuri de date.
- LargeObjects(LOB) – > imagini, video, documente randate, etc.
- Numeric – > toate tipurile de date care sunt formate din cifre.
- Character – > stocheaza un singur caracter.
- Boolean – > stocheaza valorile logice.
- DateTime – > stocheaza date calendaristice.
- Interval – > stocheaza intervale.

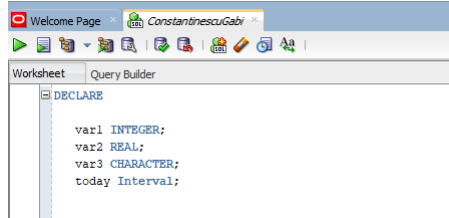


Figure 21: Exemplu cu tipuri de date

Data Type	Description
DEC, DECIMAL, or NUMERIC	Fixed-point NUMBER with maximum precision of 38 decimal digits
DOUBLE PRECISION or FLOAT	Floating-point NUMBER with maximum precision of 126 binary digits (approximately 38 decimal digits)
INT, INTEGER, or SMALLINT	Integer with maximum precision of 38 decimal digits
REAL	Floating-point NUMBER with maximum precision of 63 binary digits (approximately 18 decimal digits)
Data Type	Data Description
CHAR	Fixed-length character string with maximum size of 32,767 bytes
VARCHAR2	Variable-length character string with maximum size of 32,767 bytes
RAW	Variable-length binary or byte string with maximum size of 32,767 bytes, not interpreted by PL/SQL
NCHAR	Fixed-length national character string with maximum size of 32,767 bytes
NVARCHAR2	Variable-length national character string with maximum size of 32,767 bytes
LONG ^{Footref 1}	Variable-length character string with maximum size of 32,760 bytes
LONG RAW ^{Footref 1}	Variable-length binary or byte string with maximum size of 32,760 bytes, not interpreted by PL/SQL
ROWID ^{Footref 1}	Physical row identifier, the address of a row in an ordinary table
UROWID	Universal row identifier (physical, logical, or foreign row identifier)

Figure 22: Tabele cu tipuri de date derivate

5 Selectarea informatiilor dintr-un singur rand dintr-o tabela

Pentru a prelua informații dintr-un tabel se folosește comanda SELECT.

În momentul în care este întors un singur rând, valorile acestuia pot fi stocate în variabilele existente în blocul anonim, suprascriindu-se valorile deja existente.

Pentru a face această operație, comanda SELECT trebuie modificată pentru a include și numele variabilelor în care se face stocarea.

Se va utiliza așadar cuvântul cheie INTO pentru a face o asociere dintre valorile returnate și variabilele în care acestea vor fi stocate.

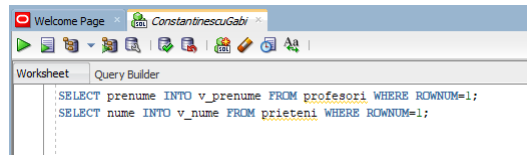


Figure 23: Exemplu de interogari

Pasii ce trebuie indeplinit pentru selectarea informatiilor dintr-un singur rand dintr-o tabela:

- Selectarea unei singure valori din tabel.

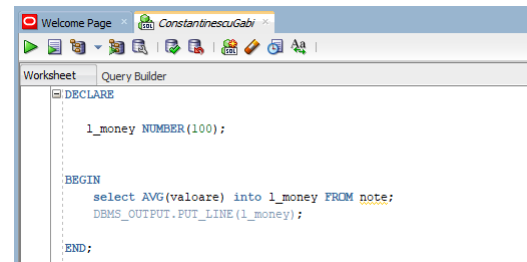


Figure 24: Exemplu

- Folosirea variabilelor cu scopul de a extrage o anumita valoare din tabel.

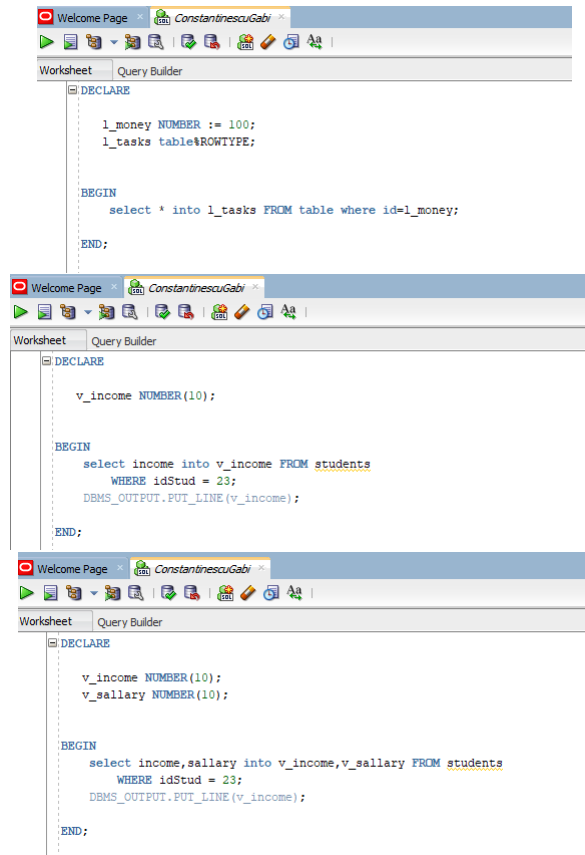


Figure 25: Exemple cu Selectarea informatiilor dintr-un singur randdintr-o tabela