

TEMĂ

În această temă voi prezenta două exemple asemănătoare cu exemplul 3.3 din curs, dar și verificarea comenzii delete cu returning.

Exemplu cu 3 Blocuri.

Să se efectueze următoarele operații în ordine:

- baza2 = baza2 la puterea baza1.
- baza1 = baza1 la puterea baza1.
- baza2 = baza2 % baza3.
- baza3 = baza3 * baza1 * baza2.

Rezolvare:

```
DECLARE
```

```
    v_baza1 INTEGER;
```

```
BEGIN
```

```
    v_baza1 := 12;
```

```
DECLARE
```

```
    v_baza2 INTEGER := 10;
```

```
BEGIN
```

```
    v_baza2 := v_baza2**v_baza1;
```

```
    DBMS_OUTPUT.PUT_LINE(v_baza2);
```

```
    v_baza1 := v_baza1**v_baza1;
```

```
    DBMS_OUTPUT.PUT_LINE(v_baza1);
```

```
DECLARE
```

```
    v_baza3 INTEGER := 7;
```

```
BEGIN
```

```
    v_baza2 := mod(v_baza2, v_baza3);
```

```
    DBMS_OUTPUT.PUT_LINE(v_baza2);
```

```
v_baza3 := v_baza3 * v_baza1 * v_baza2;
```

```
DBMS_OUTPUT.PUT_LINE(v_baza3);
```

```
END;
```

```
END;
```

```
END;
```

Print-Screen:

```
1 DECLARE
2   v_baza1 INTEGER;
3 BEGIN
4   v_baza1 := 12;
5   DECLARE
6     v_baza2 INTEGER := 10;
7   BEGIN
8
9     v_baza2 := v_baza2**v_baza1;
10    DBMS_OUTPUT.PUT_LINE(v_baza2);
11    v_baza1 := v_baza1**v_baza1;
12    DBMS_OUTPUT.PUT_LINE(v_baza1);
13    DECLARE
14      v_baza3 INTEGER := 7;
15    BEGIN
16      v_baza2 := mod(v_baza2, v_baza3);
17      DBMS_OUTPUT.PUT_LINE(v_baza2);
18      v_baza3 := v_baza3 * v_baza1 * v_baza2;
19      DBMS_OUTPUT.PUT_LINE(v_baza3);
20    END;
21  END;
22 END;
```

PL/SQL procedure successfully completed.

1000000000000
8916100448256
1

Încercăm să modificăm baza3 în interiorul blocului baza2 (baza2=baza2-baza3).

```
1 DECLARE
2   v_baza1 INTEGER;
3 BEGIN
4   v_baza1 := 12;
5   DECLARE
6     v_baza2 INTEGER := 10;
7   BEGIN
8
9     v_baza2 := v_baza2**v_baza1;
10    DBMS_OUTPUT.PUT_LINE(v_baza2);
11    v_baza1 := v_baza1**v_baza1;
12    DBMS_OUTPUT.PUT_LINE(v_baza1);
13    DECLARE
14      v_baza3 INTEGER := 7;
15    BEGIN
16      v_baza2 := mod(v_baza2, v_baza3);
17      DBMS_OUTPUT.PUT_LINE(v_baza2);
18      v_baza3 := v_baza3 * v_baza1 * v_baza2;
19      DBMS_OUTPUT.PUT_LINE(v_baza3);
20    END;
21    v_baza2 := v_baza2 - v_baza3;
22  END;
23 END;
```

Error report -
ORA-06550: line 21, column 30:
PLS-00201: identifier 'V_BAZA3' must be declared
ORA-06550: line 21, column 9:
PL/SQL: Statement ignored
06550. 00000 - "line %s, column %s:\n%s"
*Cause: Usually a PL/SQL compilation error.
*Action:

Observăm că nu putem modifica variabila baza2 întrucât variabila baza3 nu a fost declarată. Ea a fost declarată local în blocul baza3 deci, odată ce s-a terminat blocul baza3, nu mai poate fi accesibilă (a ieșit din scope, primim eroare de compilare).

Exemplu cu 4 Blocuri.

- Să se afișeze variabila din blocul 1 în interiorul blocului 1.
- Să se afișeze variabila din blocul 1 în interiorul blocului 2.
- Să se afișeze variabila din blocul 2 în interiorul blocului 2.
- Să se calculeze și să se afișeze suma dintre blocul 1 și blocul 2, în interiorul blocului 2.
- Să se calculeze și să se afișeze diferența dintre blocul 1 și blocul 2, în interiorul blocului 2.
- Să se afișeze variabila din blocul 1 în interiorul blocului 3.
- Să se afișeze variabila din blocul 2 în interiorul blocului 3.
- Să se afișeze variabila din blocul 3 în interiorul blocului 3.
- Să se calculeze și să se afișeze câtul împărțirii **bloc2/(bloc1/bloc3)**, în interiorul blocului 3.
- Să se afișeze variabila din blocul 1 în interiorul blocului 4.
- Să se afișeze variabila din blocul 2 în interiorul blocului 4.
- Să se afișeze variabila din blocul 3 în interiorul blocului 4.
- Să se afișeze variabila din blocul 4 în interiorul blocului 4.
- Să se calculeze și să se afișeze restul împărțirii **bloc4%(bloc1/bloc3)**, în interiorul blocului 4.
- Să se modifice valoarea din blocul 1 în interiorul blocului 4 cu 121.
- Să se modifice valoarea din blocul 2 în interiorul blocului 4 cu 11.
- Să se modifice valoarea din blocul 2 în interiorul blocului 4 cu 7.
- Să se modifice valoarea din blocul 2 în interiorul blocului 4 cu 3.

Rezolvare:

DECLARE

v_nume_bloc_1 VARCHAR2(30);

v_bloc_1 NUMBER(10);

v_suma NUMBER(10) := 0;

v_diferenta NUMBER(10) := 0;

v_vizitat VARCHAR2(100);

BEGIN

v_nume_bloc_1 := 'PRINCIPAL';

v_bloc_1 := 64;

DBMS_OUTPUT.PUT_LINE('!!! AM INTRAT IN BLOCUL ' || v_nume_bloc_1 || ' !!!');

```

    DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_num_bloc_1 || ' in blocul ' ||
v_num_bloc_1 || ', care are valoarea ' || v_bloc_1 || '.');

DECLARE

    v_num_bloc_2 VARCHAR2(30);

    v_bloc_2 NUMBER(10);

BEGIN

    v_num_bloc_2 := 'SECUNDAR';

    v_bloc_2 := 8;

    DBMS_OUTPUT.PUT_LINE('!!! AM INTRAT IN BLOCUL ' || v_num_bloc_2 || ' !!!');

    DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_num_bloc_1 || ' in blocul ' ||
v_num_bloc_2 || ', care are valoarea ' || v_bloc_1 || '.');

    DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_num_bloc_2 || ' in blocul ' ||
v_num_bloc_2 || ', care are valoarea ' || v_bloc_2 || '.');

    DBMS_OUTPUT.PUT_LINE('->Calculez suma dintre blocul ' || v_num_bloc_1 || ' si cel ' ||
v_num_bloc_2 || '.');

    v_suma := v_bloc_1 + v_bloc_2;

    DBMS_OUTPUT.PUT_LINE('Suma este: ' || v_suma || '.');

    DBMS_OUTPUT.PUT_LINE('->Calculez diferenta dintre blocul ' || v_num_bloc_1 || ' si cel ' ||
v_num_bloc_2 || '.');

    v_diferenta := v_bloc_1 - v_bloc_2;

    DBMS_OUTPUT.PUT_LINE('Diferenta este: ' || v_diferenta || '.');

DECLARE

    v_num_bloc_3 VARCHAR2(30);

    v_bloc_3 NUMBER(10);

    v_impirtire NUMBER(10) := 0;

BEGIN

    v_num_bloc_3 := 'SECUNDAR2';

    v_bloc_3 := 16;

    DBMS_OUTPUT.PUT_LINE('!!! AM INTRAT IN BLOCUL ' || v_num_bloc_3 || ' !!!');

```

```
DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_nume_bloc_1 || ' in blocul ' ||  
v_nume_bloc_3 || ', care are valoarea ' || v_bloc_1 || '.');
```

```
DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_nume_bloc_2 || ' in blocul ' ||  
v_nume_bloc_3 || ', care are valoarea ' || v_bloc_2 || '.');
```

```
DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_nume_bloc_3 || ' in blocul ' ||  
v_nume_bloc_3 || ', care are valoarea ' || v_bloc_3 || '.');
```

```
DBMS_OUTPUT.PUT_LINE('->Calculez catul impartirii: ' || v_nume_bloc_2 || '/' ||  
v_nume_bloc_1 || '/' || v_nume_bloc_3 || '.');
```

```
v_impartire := v_bloc_2 / (v_bloc_1/v_bloc_3);
```

```
DBMS_OUTPUT.PUT_LINE('Catul este: ' || v_impartire || '.');
```

```
DECLARE
```

```
v_nume_bloc_4 VARCHAR2(30);
```

```
v_bloc_4 NUMBER(10);
```

```
v_rest NUMBER(10) := 0;
```

```
BEGIN
```

```
v_nume_bloc_4 := 'SECUNDAR3';
```

```
v_bloc_4 := 5;
```

```
DBMS_OUTPUT.PUT_LINE('!!! AM INTRAT IN BLOCUL ' || v_nume_bloc_4 || ' !!!');
```

```
DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_nume_bloc_1 || ' in blocul ' ||  
v_nume_bloc_4 || ', care are valoarea ' || v_bloc_1 || '.');
```

```
DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_nume_bloc_2 || ' in blocul ' ||  
v_nume_bloc_4 || ', care are valoarea ' || v_bloc_2 || '.');
```

```
DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_nume_bloc_3 || ' in blocul ' ||  
v_nume_bloc_4 || ', care are valoarea ' || v_bloc_3 || '.');
```

```
DBMS_OUTPUT.PUT_LINE('Accesez variabila din blocul ' || v_nume_bloc_4 || ' in blocul ' ||  
v_nume_bloc_4 || ', care are valoarea ' || v_bloc_4 || '.');
```

```
DBMS_OUTPUT.PUT_LINE('->Calculez restul impartirii: ' || v_nume_bloc_4 || '%' ||  
v_nume_bloc_1 || '/' || v_nume_bloc_3 || '.');
```

```
v_rest := mod(v_bloc_4, v_bloc_1/v_bloc_3);
```

```
DBMS_OUTPUT.PUT_LINE('Restul este: ' || v_rest || '.');
```

```

        DBMS_OUTPUT.PUT_LINE('Incercam sa dam valoarea 121 variabilei din blocul ' ||
v_nume_bloc_1 || '.');

        v_bloc_1 := 121;

        DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_nume_bloc_1 || ' are valoarea acum: ' ||
v_bloc_1 || '.');

        DBMS_OUTPUT.PUT_LINE('Incercam sa dam valoarea 11 variabilei din blocul ' ||
v_nume_bloc_2 || '.');

        v_bloc_2 := 11;

        DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_nume_bloc_2 || ' are valoarea acum: ' ||
v_bloc_2 || '.');

        DBMS_OUTPUT.PUT_LINE('Incercam sa dam valoarea 7 variabilei din blocul ' || v_nume_bloc_3
|| '.');

        v_bloc_3 := 7;

        DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_nume_bloc_3 || ' are valoarea acum: ' ||
v_bloc_3 || '.');

        DBMS_OUTPUT.PUT_LINE('Incercam sa dam valoarea 3 variabilei din blocul ' || v_nume_bloc_4
|| '.');

        v_bloc_4 := 3;

        DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_nume_bloc_4 || ' are valoarea acum: ' ||
v_bloc_4 || '.');

        DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_nume_bloc_4 || ' !!!');

        v_vizitat := v_nume_bloc_4;

END;

        DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_nume_bloc_3 || ' !!!');

        v_vizitat := v_vizitat || ', ' || v_nume_bloc_3;

END;

        DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_nume_bloc_2 || ' !!!');

        v_vizitat := v_vizitat || ', ' || v_nume_bloc_2;

END;

        DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_nume_bloc_1 || ' !!!');

```

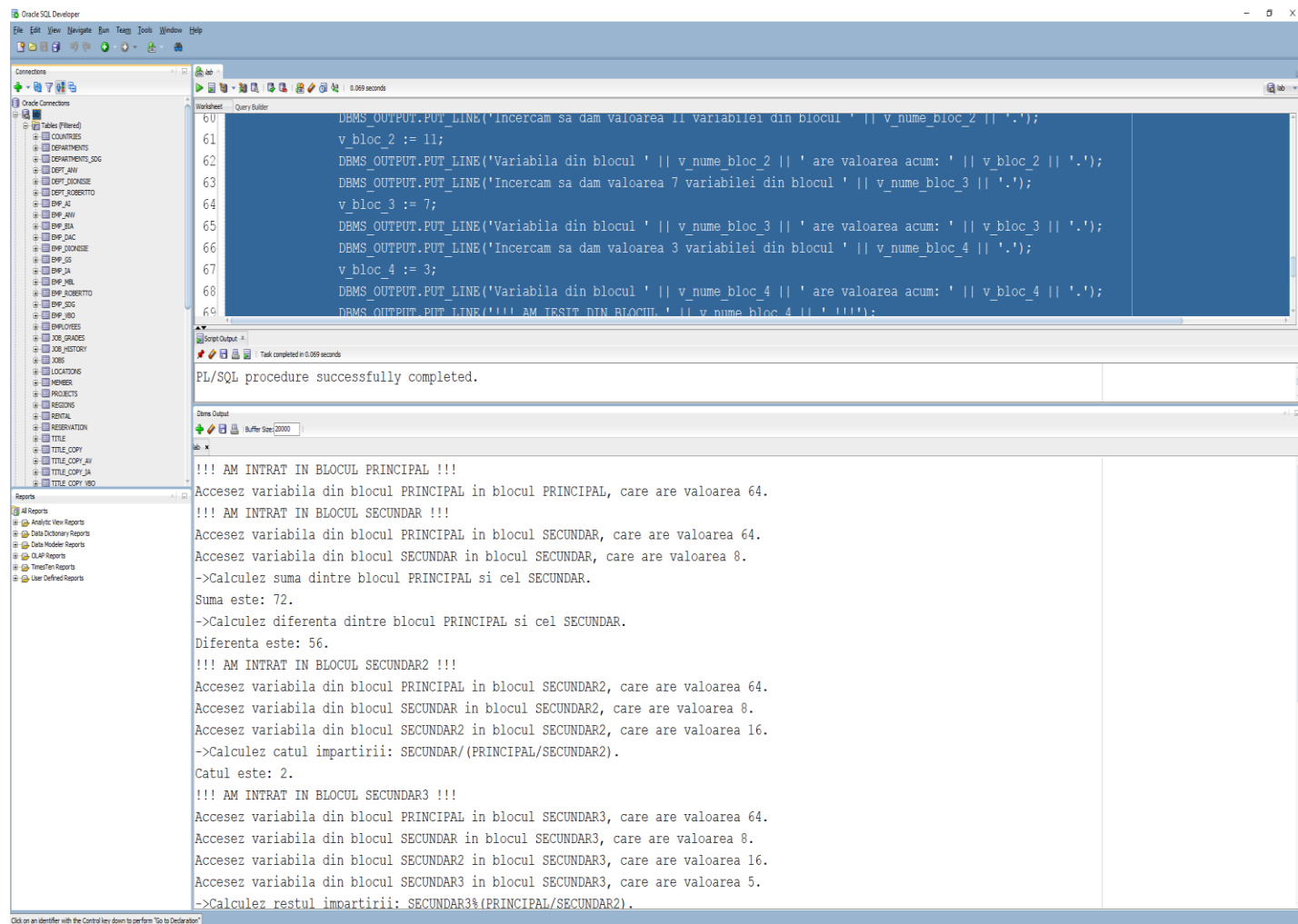
```
v_vizitat := v_vizitat || ',' || v_num_bloc_1;
```

```
DBMS_OUTPUT.PUT_LINE('CONCLUZIE! ORDINEA IN CARE S-AU TERMINAT DE PARCURS BLOCURILE  
ESTE: ' || v_vizitat || '.');
```

```
DBMS_OUTPUT.PUT_LINE('CE OBSERVAM? OBSERVAM CA BLOCURILE SE TERMINA IN ORDINEA  
DESCRESCATOARE IN CARE S-AU CREAT ELE!');
```

```
END;
```

Print-Screen:



The screenshot displays the Oracle SQL Developer interface. The 'Connections' pane on the left shows the 'hr' schema selected. The 'Query Builder' pane in the center contains a PL/SQL script. The 'Script Output' pane at the bottom shows the execution results, indicating that the PL/SQL procedure was successfully completed. The output text is as follows:

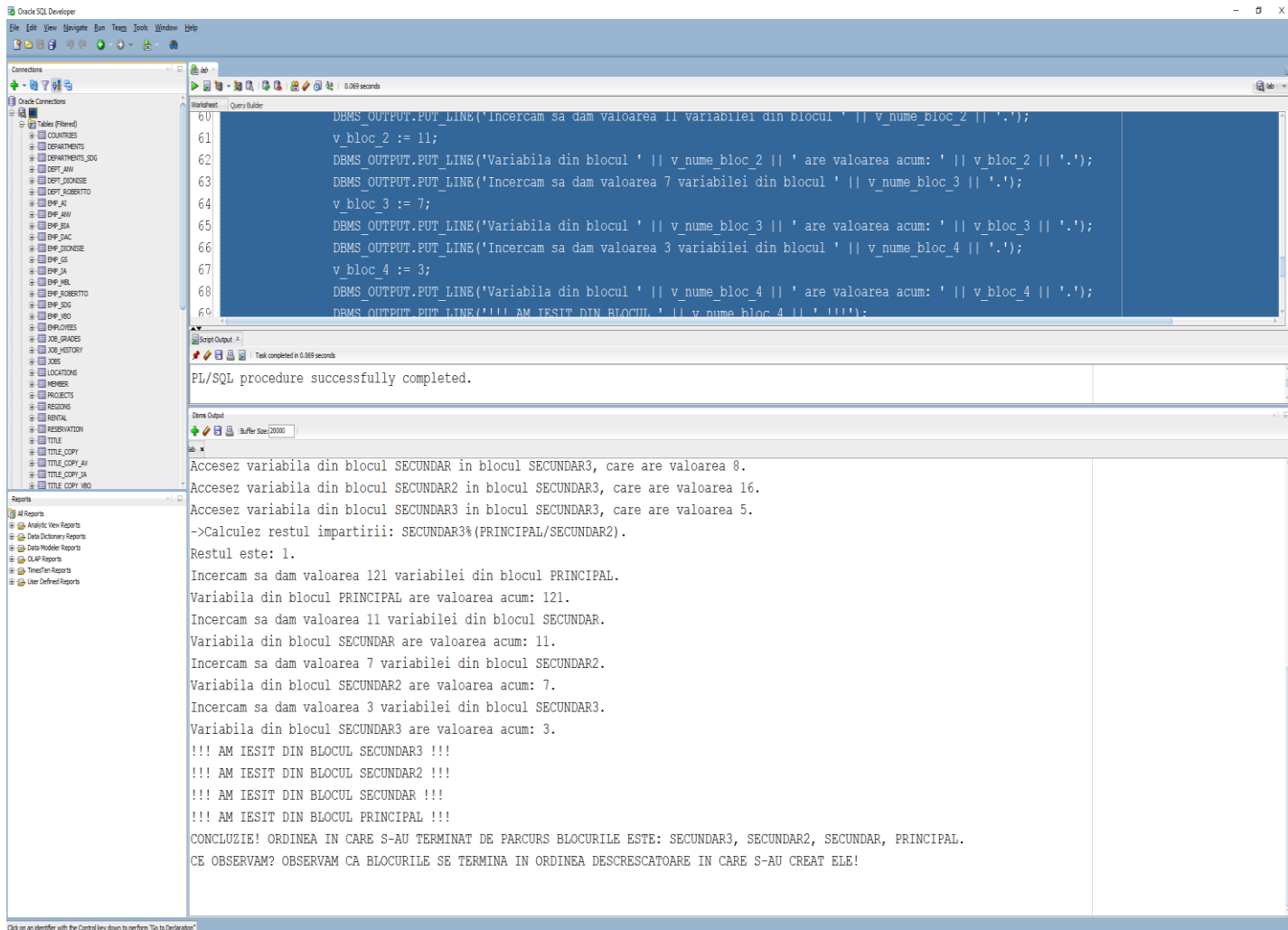
```
60 DBMS_OUTPUT.PUT_LINE('Inceram sa dam valoarea 11 variabilei din blocul ' || v_num_bloc_2 || '.');
61 v_bloc_2 := 11;
62 DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_num_bloc_2 || ' are valoarea acum: ' || v_bloc_2 || '.');
63 DBMS_OUTPUT.PUT_LINE('Inceram sa dam valoarea 7 variabilei din blocul ' || v_num_bloc_3 || '.');
64 v_bloc_3 := 7;
65 DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_num_bloc_3 || ' are valoarea acum: ' || v_bloc_3 || '.');
66 DBMS_OUTPUT.PUT_LINE('Inceram sa dam valoarea 3 variabilei din blocul ' || v_num_bloc_4 || '.');
67 v_bloc_4 := 3;
68 DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_num_bloc_4 || ' are valoarea acum: ' || v_bloc_4 || '.');
69 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_4 || ' !!!');
```

Task completed in 0.069 seconds.

PL/SQL procedure successfully completed.

Script Output

!!! AM INTRAT IN BLOCUL PRINCIPAL !!!
Accesez variabila din blocul PRINCIPAL in blocul PRINCIPAL, care are valoarea 64.
!!! AM INTRAT IN BLOCUL SECUNDAR !!!
Accesez variabila din blocul PRINCIPAL in blocul SECUNDAR, care are valoarea 64.
Accesez variabila din blocul SECUNDAR in blocul SECUNDAR, care are valoarea 8.
->Calculez suma dintre blocul PRINCIPAL si cel SECUNDAR.
Suma este: 72.
->Calculez diferenta dintre blocul PRINCIPAL si cel SECUNDAR.
Diferenta este: 56.
!!! AM INTRAT IN BLOCUL SECUNDAR2 !!!
Accesez variabila din blocul PRINCIPAL in blocul SECUNDAR2, care are valoarea 64.
Accesez variabila din blocul SECUNDAR in blocul SECUNDAR2, care are valoarea 8.
Accesez variabila din blocul SECUNDAR2 in blocul SECUNDAR2, care are valoarea 16.
->Calculez catul impartirii: SECUNDAR/(PRINCIPAL/SECUNDAR2).
Catul este: 2.
!!! AM INTRAT IN BLOCUL SECUNDAR3 !!!
Accesez variabila din blocul PRINCIPAL in blocul SECUNDAR3, care are valoarea 64.
Accesez variabila din blocul SECUNDAR in blocul SECUNDAR3, care are valoarea 8.
Accesez variabila din blocul SECUNDAR2 in blocul SECUNDAR3, care are valoarea 16.
Accesez variabila din blocul SECUNDAR3 in blocul SECUNDAR3, care are valoarea 5.
->Calculez restul impartirii: SECUNDAR3%(PRINCIPAL/SECUNDAR2).



➤ Ce observăm?

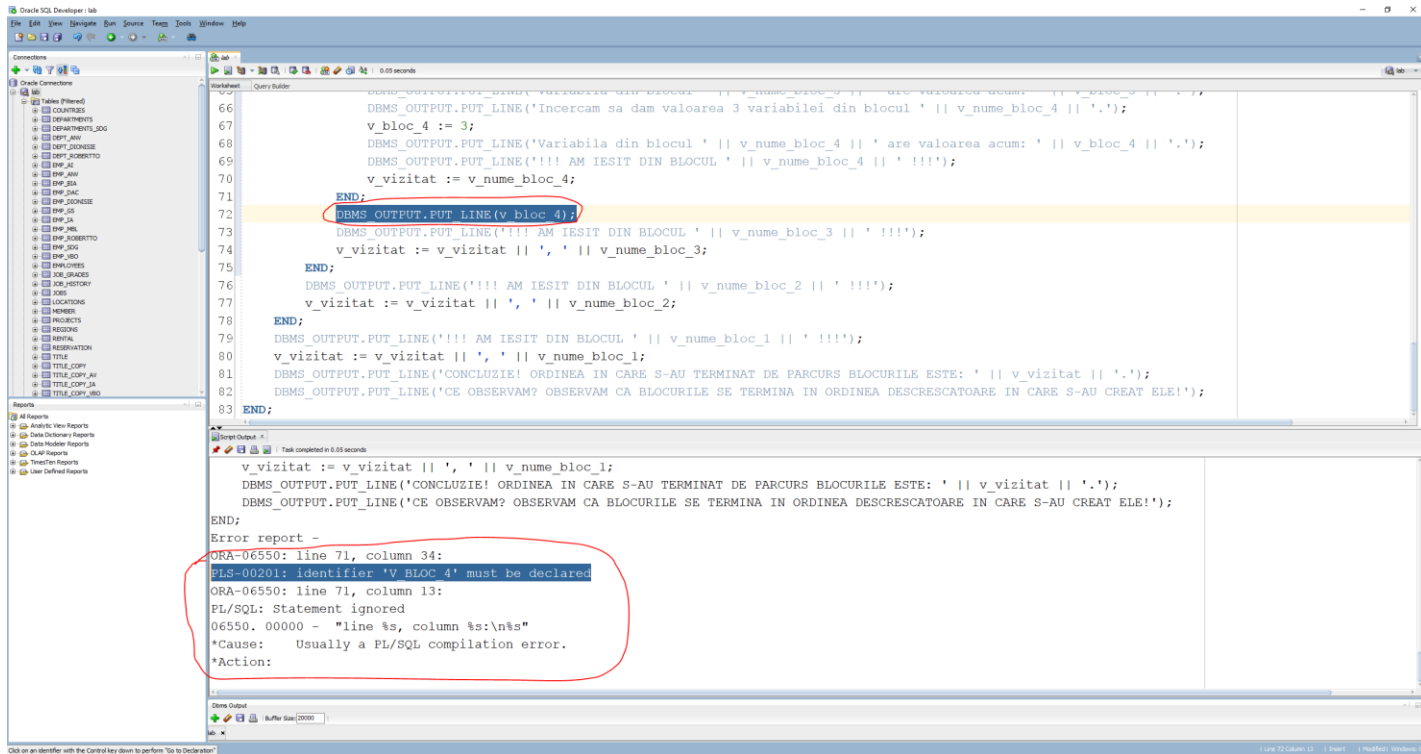
Observăm că variabilele din blocurile părinte sunt accesibile în blocurile de tip fii. Ce înseamnă asta?

Totul funcționează pe un principiu ierarhic. În exemplul nostru blocul 1 este tatăl blocului 2, care la rândul lui tatăl blocului 3, care la rândul lui este tatăl blocului 4.

Deci în interiorul blocului 4 putem accesa toate datele din blocurile mai mari ca el (**bloc3, bloc2, bloc1**), fiind fiul cel mai mic, în interiorul blocului 3 putem accesa toate datele din blocurile mai mari ca el (**bloc2, bloc1**), fiind tatăl blocului 4 și fiul blocului 2. !!! Nu putem accesa datele din blocul fiu (**bloc4**)!!!. În interiorul blocului 2 putem accesa doar datele din **blocul1**, fiind tatăl său, iar datele din descendenții săi sunt inaccesibile (**bloc3, bloc4**). În interiorul blocului 1 nu mai putem accesa alte date în afară de cele din blocul1.

De ce se întâmplă acest lucru? Pentru că în fiecare bloc avem variabile declarate local! Cum am fi putut accesa orice variabile de oriunde? Declaram totul global.

Testăm accesarea variabilei din blocul 4 în blocul 3:

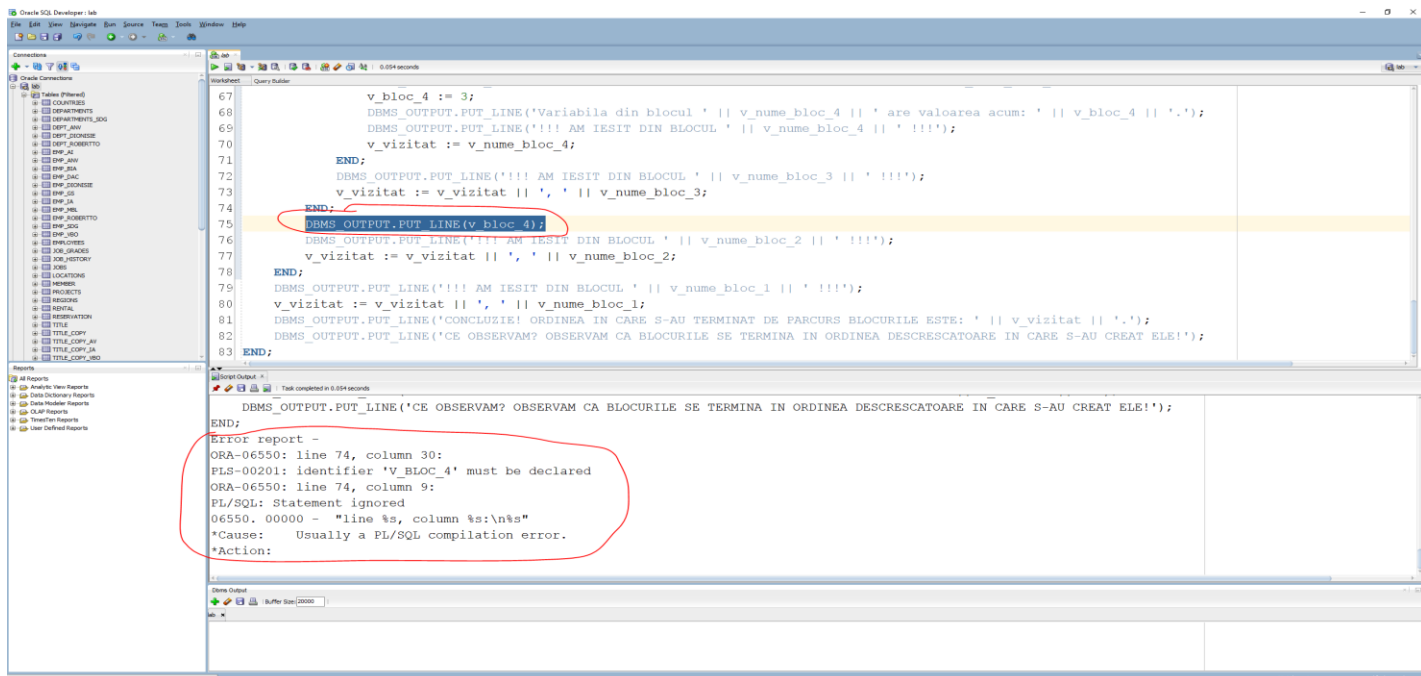


```
66 DBMS_OUTPUT.PUT_LINE('Incercam sa dam valoarea 3 variabilei din blocul ' || v_num_bloc_4 || '.');
67 v_bloc_4 := 3;
68 DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_num_bloc_4 || ' are valoarea acum: ' || v_bloc_4 || '.');
69 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_4 || ' !!!');
70 v_vizitat := v_num_bloc_4;
71 END;
72 DBMS_OUTPUT.PUT_LINE(v_bloc_4);
73 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_3 || ' !!!');
74 v_vizitat := v_vizitat || ', ' || v_num_bloc_3;
75 END;
76 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_2 || ' !!!');
77 v_vizitat := v_vizitat || ', ' || v_num_bloc_2;
78 END;
79 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_1 || ' !!!');
80 v_vizitat := v_vizitat || ', ' || v_num_bloc_1;
81 DBMS_OUTPUT.PUT_LINE('CONCLUZIE! ORDINEA IN CARE S-AU TERMINAT DE PARCURS BLOCURILE ESTE: ' || v_vizitat || '.');
82 DBMS_OUTPUT.PUT_LINE('CE OBSERVAM? OBSERVAM CA BLOCURILE SE TERMINA IN ORDINEA DESCRESATOARE IN CARE S-AU CREAT ELE!');
83 END;
```

Error report -
ORA-06550: line 71, column 34:
PLS-00201: identifier 'V_BLOC_4' must be declared
ORA-06550: line 71, column 13:
PL/SQL: Statement ignored
06550. 00000 - "line %s, column %s:\n%s"
*Cause: Usually a PL/SQL compilation error.
*Action:

Această eroare ne spune că variabila “**v_bloc_4**” nu a fost declarată, întrucât ea a fost declarată local în interiorul blocului 4. Odată ce am ieșit din blocul 4 (din scope), ea nu mai este accesibilă întrucât ea a fost declarată local. (**EROARE DE COMPILARE!**)

Testăm accesarea variabilei din blocul 4 în blocul 2:



```
67 v_bloc_4 := 3;
68 DBMS_OUTPUT.PUT_LINE('Variabila din blocul ' || v_num_bloc_4 || ' are valoarea acum: ' || v_bloc_4 || '.');
69 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_4 || ' !!!');
70 v_vizitat := v_num_bloc_4;
71 END;
72 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_3 || ' !!!');
73 v_vizitat := v_vizitat || ', ' || v_num_bloc_3;
74 END;
75 DBMS_OUTPUT.PUT_LINE(v_bloc_4);
76 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_2 || ' !!!');
77 v_vizitat := v_vizitat || ', ' || v_num_bloc_2;
78 END;
79 DBMS_OUTPUT.PUT_LINE('!!! AM IESIT DIN BLOCUL ' || v_num_bloc_1 || ' !!!');
80 v_vizitat := v_vizitat || ', ' || v_num_bloc_1;
81 DBMS_OUTPUT.PUT_LINE('CONCLUZIE! ORDINEA IN CARE S-AU TERMINAT DE PARCURS BLOCURILE ESTE: ' || v_vizitat || '.');
82 DBMS_OUTPUT.PUT_LINE('CE OBSERVAM? OBSERVAM CA BLOCURILE SE TERMINA IN ORDINEA DESCRESATOARE IN CARE S-AU CREAT ELE!');
83 END;
```

Error report -
ORA-06550: line 74, column 30:
PLS-00201: identifier 'V_BLOC_4' must be declared
ORA-06550: line 74, column 9:
PL/SQL: Statement ignored
06550. 00000 - "line %s, column %s:\n%s"
*Cause: Usually a PL/SQL compilation error.
*Action:

Observăm că se întâmplă același lucru. Variabila nu este accesibilă întrucât nu a fost declarată (a ieșit din scope-ul ei, respectiv **baza4**), ceea ce duce la o eroare de compilare.

Concluzie:

- Blocurile se termină în ordinea descrescătoare în care s-au creat (se creează bloc1, bloc2, bloc3 se va termina bloc3, după bloc2, după bloc1)
- Variabilele declarate într-un bloc sunt declarate local și sunt accesibile doar în blocurile sale descendente.
- Dacă încercăm să modificăm/accesăm o variabilă dintr-un bloc descendent, iar noi ne aflăm în afara acestui bloc, vom primi eroare de compilare când vom rula codul în SQL Developer (**PLS-00201: identifier 'nume_variabila' must be declared**).

Verficarea comenzii delete cu returning.

DECLARE

v_emp_first_name VARCHAR2(30);

v_emp_last_name VARCHAR2(30);

v_emp_manager NUMBER(10) := 1337;

v_emp_salary NUMBER(30) := 1337;

BEGIN

DELETE FROM emp_Robertto WHERE manager_id = 101

RETURNING first_name, last_name

INTO v_emp_first_name, v_emp_last_name;

DBMS_OUTPUT.PUT_LINE ('<<DELETE RETURNING INTO...>>');

DBMS_OUTPUT.PUT_LINE ('First Name: ' || v_emp_first_name);

DBMS_OUTPUT.PUT_LINE ('Last Name: ' || v_emp_last_name);

UPDATE emp_Robertto

SET salary = v_emp_salary

WHERE employee_id = 0;

```

UPDATE emp_Robertto
SET salary = v_emp_salary
WHERE employee_id = 0
RETURNING first_name, last_name, salary
INTO v_emp_first_name, v_emp_last_name, v_emp_salary;
DBMS_OUTPUT.PUT_LINE ('<<UPDATE RETURNING INTO...>>');
DBMS_OUTPUT.PUT_LINE ('First Name: ' || v_emp_first_name);
DBMS_OUTPUT.PUT_LINE ('Last Name: ' || v_emp_last_name);
DBMS_OUTPUT.PUT_LINE ('Salary: ' || v_emp_salary);
COMMIT;
EXCEPTION
WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('SE RETURNEAZA MAI MULT DE 1 LINIE!');
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('NU A FOST GASIT UN ASEMENEA ANGAJAT!');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ALTA EROARE!');
END;

```

Print-Screen:

Vom testa stergerea mai multor linii cu returning into variabile.

În poza de mai jos putem observa ca avem 5 înregistrări pentru care manager_id = 101;

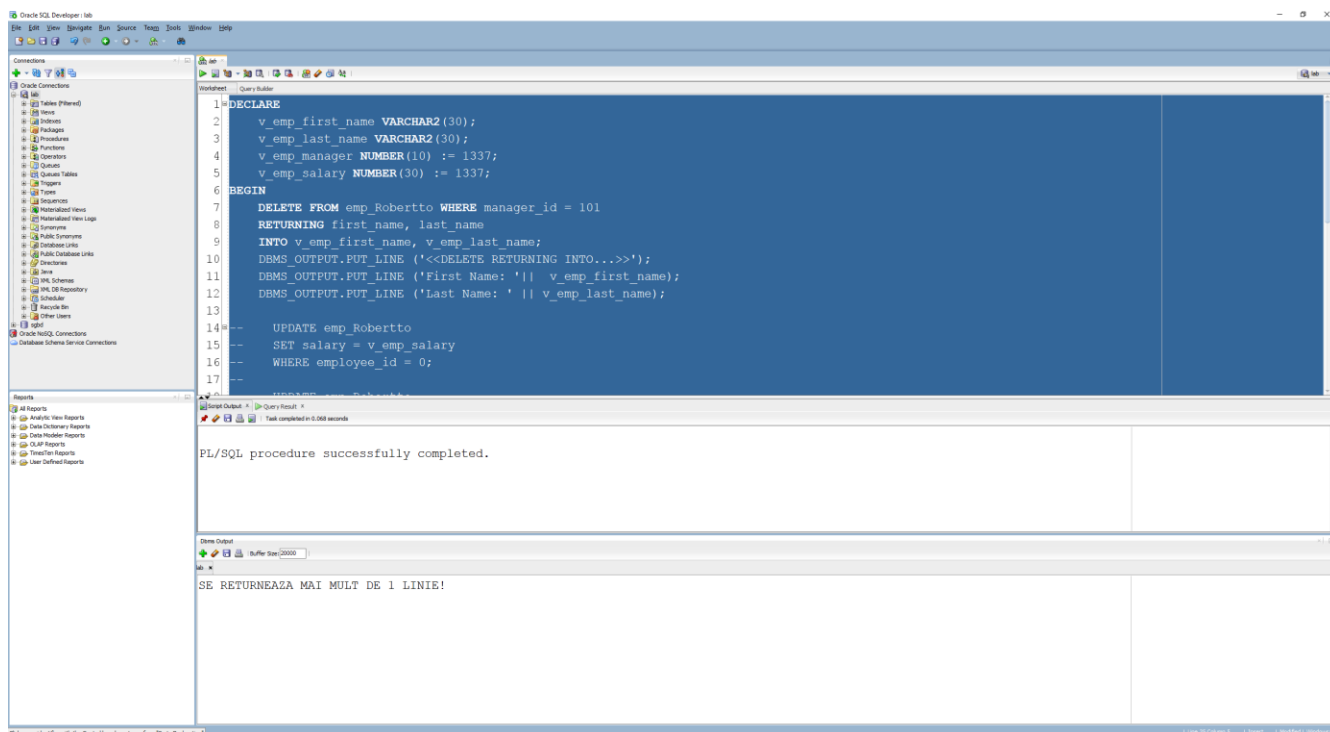
```

37 select * from emp_Robertto where manager_id=101;
38 select * from emp_Robertto where employee_id=103;
39 select * from emp_Robertto where employee_id=0;
40 rollback;

```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD ASST	4400	(null)	101	10
2	203	Susan	Mavris	SMAVRIS	515.123.7777	07-JUN-94	HR REP	6500	(null)	101	40
3	204	Hermann	Baer	HBAER	515.123.8888	07-JUN-94	PR REP	10000	(null)	101	70
4	205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC MGR	12000	(null)	101	110
5	108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-94	FI MGR	12000	(null)	101	100

Încercăm să ștergem toate înregistrările pentru care `manager_id = 101`; și să le reținem datele în variabile (`v_emp_first_name` și `v_emp_last_name`).



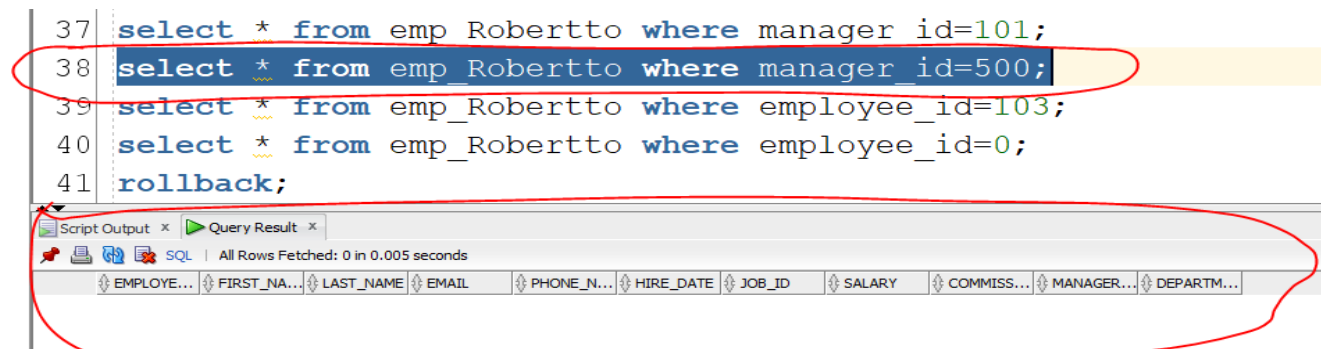
➤ Ce observăm?

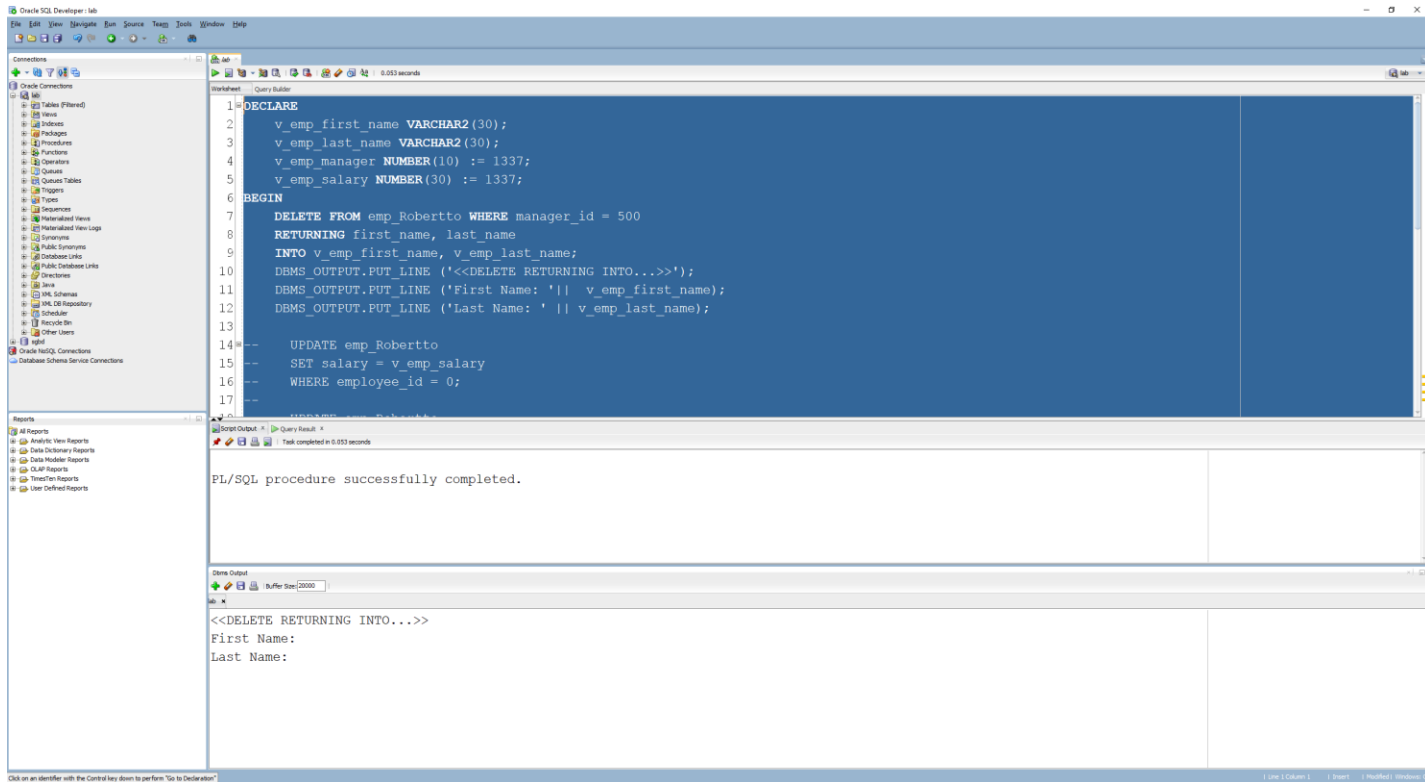
Observăm că intră pe excepția **TOO_MANY_ROWS**. De ce? Nu poate stoca mai multe valori într-o variabilă. Noi returnăm 5 înregistrări, și nu le putem stoca pe toate într-o singură variabilă.

Dacă nu am fi vrut să reținem înregistrările pe care vrem să le ștergem în variabile totul ar fi mers cu succes:

```
DELETE FROM emp_Robertto WHERE manager_id = 101;
```

Acum încercăm pentru cazul în care sunt 0 înregistrări de șters de salvat în variabile. Știm că nu există niciun angajat cu `manager_id = 500`; Deci vom testa pentru acest caz pentru a vedea dacă programul va intra pe excepția **NO_DATA_FOUND**.





➤ **Ce observăm?**

Observăm că nu intră pe excepția **NO_DATA_FOUND**. De ce? Pentru că nu există nicio înregistrare pentru care manager_id = 500; deci noi nu salvăm nicio înregistrare în variabilele noastre. Practic ele rămân cu valoare default cu care au fost declarate.

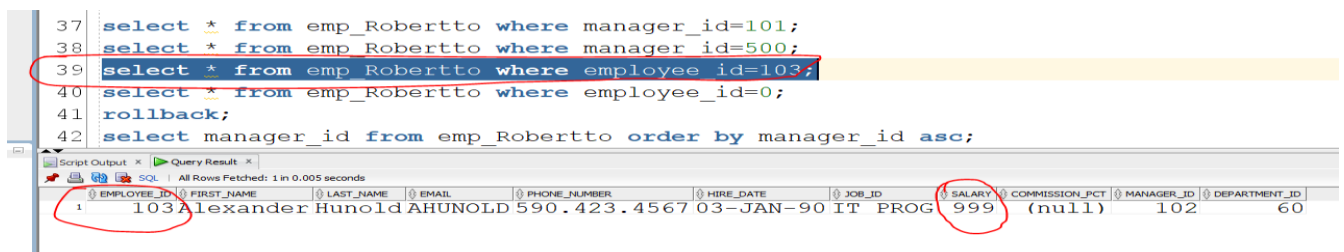
Dacă am fi declarat în zona **DECLARE** variabilele: `v_emp_first_name := 'Gigel'` și `v_emp_last_name := 'Cornel'`. Pe ecran s-ar fi afișat:

<<DELETE RETURNING INTO...>>

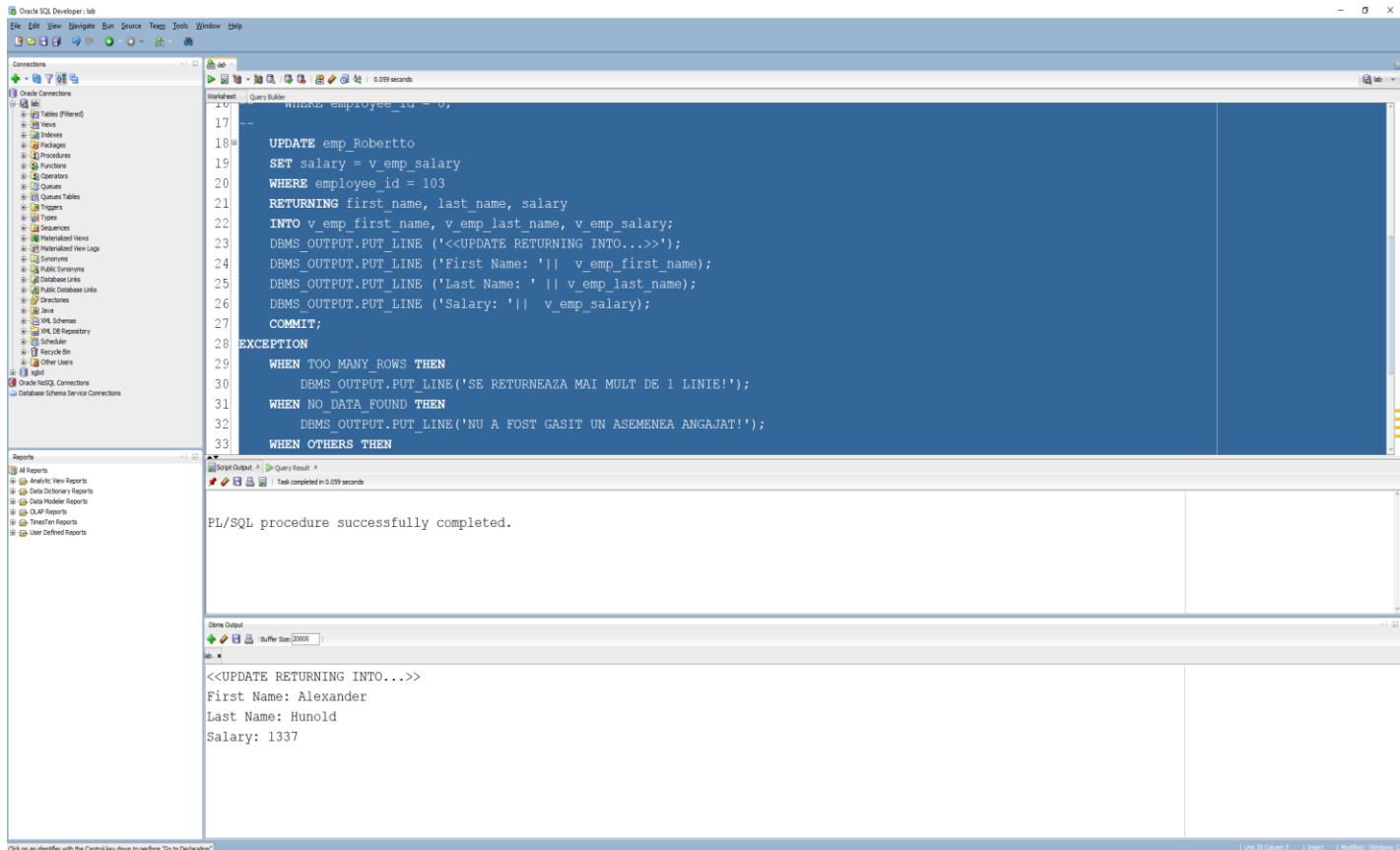
First Name: Gigel

Last Name: Cornel

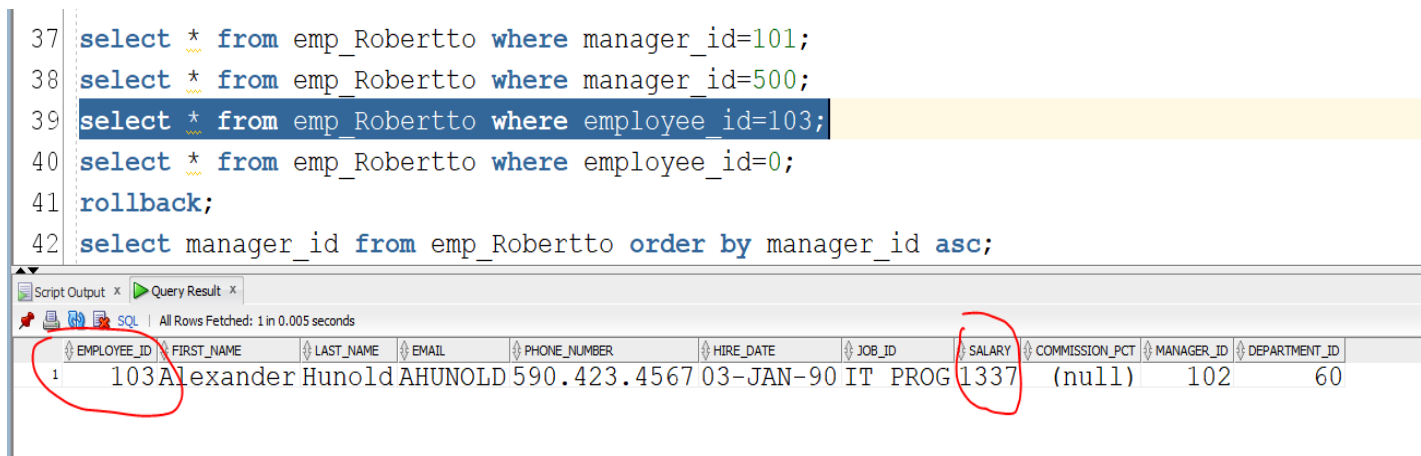
Acum testăm pentru comanda **UPDATE**. Încercăm să updatăm linia pentru care employee_id = 103;



Avem o singură înregistrare, deci totul ar trebui să funcționeze cu succes.

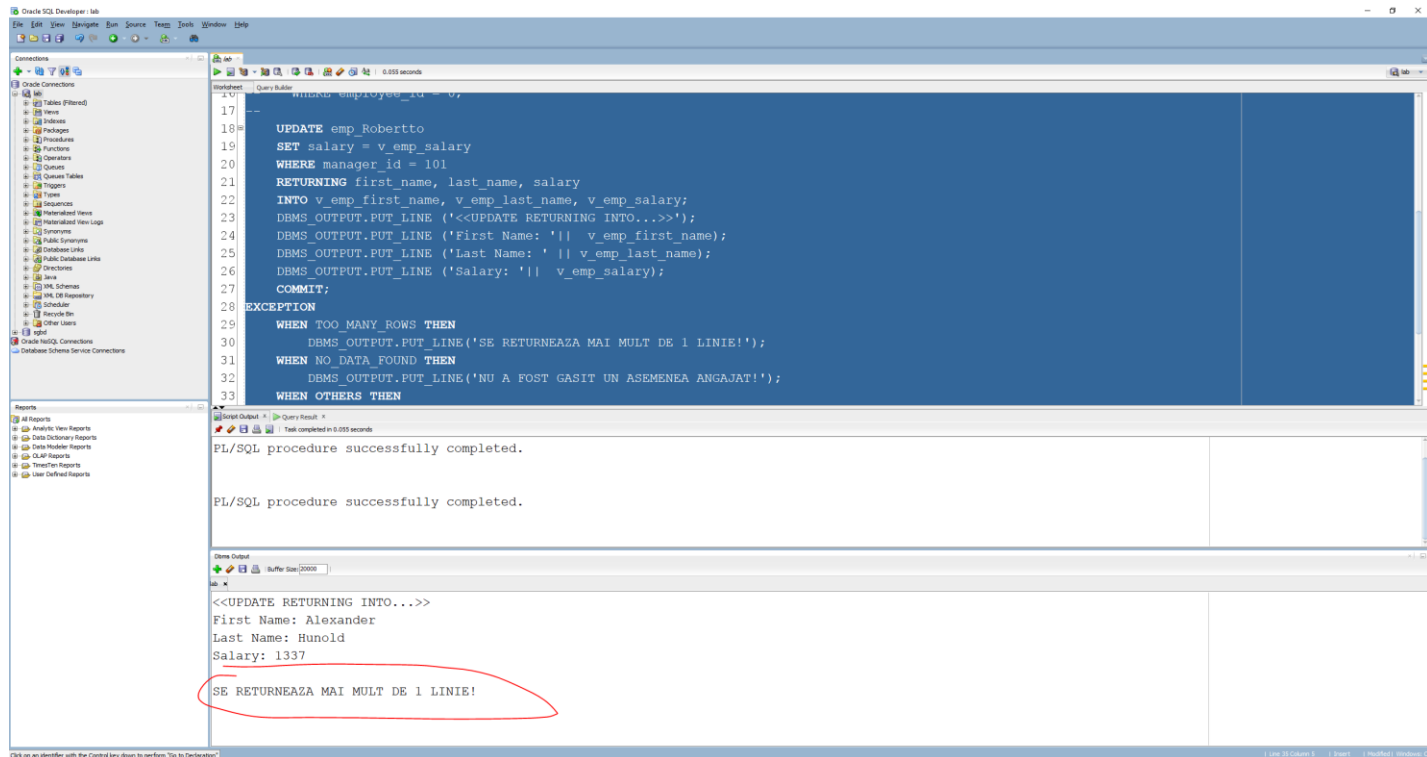


A funcționat, acum verificăm dacă s-a actualizat și în tabel.



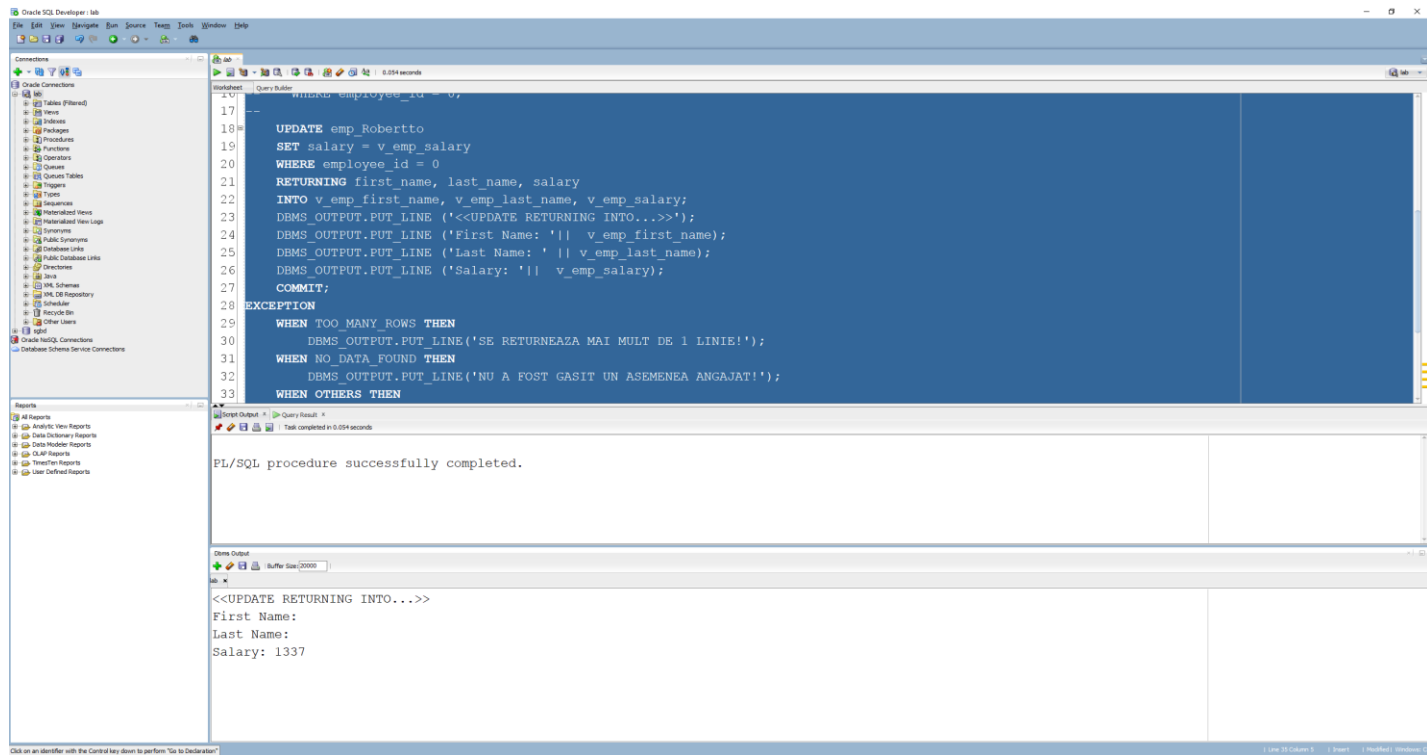
Totul a funcționat cu succes.

Acum verificăm din nou pentru cazul când avem mai multe linii (pentru manager_id = 101).



Intră de data aceasta pe excepția **TOO_MANY_ROWS**.

Încercăm cu 0 înregistrări (pentru employee_id = 0).



OBSERVAȚIE! SE COMPORTĂ LA FEL CA DELETE RETURNING INTO...

Concluzie:

- Dacă avem **mai mult de o înregistrare** care satisface un criteriu, atunci când vrem să folosim **RETURNING INTO** pentru a le salva datele în variabile, **VOM PRIMI EROAREA TOO_MANY_ROWS**, care poate fi tratată cu o excepție.
- Dacă avem **o singură înregistrare** totul se va desfășura cu succes.
- Dacă nu avem nicio înregistrare care satisface un criteriu, atunci când vrem să folosim **RETURNING INTO** pentru a salva datele în variabile, **NU VOM PRIMI EROAREA NO_DATA_FOUND !**, pentru că practic *“noi salvăm nimic”* în acele variabile, ele rămânând cu valorile default, din zona **DECLARE**.

*Când intrăm pe excepția **NO_DATA_FOUND** ?*

Mai jos avem un exemplu când intrăm pe excepția **NO_DATA_FOUND**.

The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL procedure named `update_salaries` with the following code:

```
22 -- INTO v_emp_first_name, v_emp_last_name, v_emp_salary;
23 -- DBMS_OUTPUT.PUT_LINE ('<<UPDATE RETURNING INTO...>>');
24 -- DBMS_OUTPUT.PUT_LINE ('First Name: ' || v_emp_first_name);
25 -- DBMS_OUTPUT.PUT_LINE ('Last Name: ' || v_emp_last_name);
26 -- DBMS_OUTPUT.PUT_LINE ('Salary: ' || v_emp_salary);
27 SELECT first_name, last_name INTO v_emp_first_name, v_emp_last_name
28 FROM emp_Robertto
29 WHERE manager_id = v_emp_manager;
30 DBMS_OUTPUT.PUT_LINE ('First Name: ' || v_emp_first_name);
31 DBMS_OUTPUT.PUT_LINE ('Last Name: ' || v_emp_last_name);
32 COMMIT;
33 EXCEPTION
34 WHEN TOO_MANY_ROWS THEN
35   DBMS_OUTPUT.PUT_LINE('SE RETURNEAZA MAI MULT DE 1 LINIE!');
36 WHEN NO_DATA_FOUND THEN
37   DBMS_OUTPUT.PUT_LINE('NU A FOST GASIT UN ASEMENEA ANGAJAT!');
38 WHEN OTHERS THEN
```

The execution results pane shows the message: **PL/SQL procedure successfully completed.**

The SQL Output pane shows the message: **NU A FOST GASIT UN ASEMENEA ANGAJAT!**

Cauza: Nu avem niciun angajat pentru care `manager_id = v_emp_manager = 1337`, deci nu se poate îndeplini filtrarea după `manager_id`. Rezultând astfel eroarea **NO_DATA_FOUND**, care se poate trata cu o excepție (în cazul nostru am afișat pe ecran *‘NU A FOST GASIT UN ASEMENEA ANGAJAT!’*).