

Tehnici de programare cu baze de date

#9 PL/SQL Proceduri în PL/SQL (partea a II-a)

Adrian Runceanu
www.runceanu.ro/adrian

Curs 9

Proceduri în PL/SQL (partea a II-a)

Cuprins

Proceduri în PL/SQL

I. Transmiterea parametrilor în proceduri

I. Transmiterea parametrilor în proceduri

- Pentru ca procedurile sa fie mai flexibile este important ca diverse date sa fie transmise procedurii prin parametrii de intrare.
- Rezultatele determinate pot fi returnate prin folosirea parametrilor de tip **OUT** sau **IN OUT**.

I. Transmiterea parametrilor în proceduri

Tipuri de parametri procedurali

Tipurile parametrilor sunt specificate în declararea parametrilor formali, după numele parametrului și înainte de tipul sau de date.

I. Transmiterea parametrilor în proceduri

Tipurile parametrilor sunt:

1. parametru de tip **IN** (implicit) – *furnizeaza date de intrare subprogramului*
2. parametru de tip **OUT** – *returneaza rezultate de la subprogram*
3. parametru de tip **IN OUT** – *furnizeaza o valoare de intrare care poate fi returnata modificata.*

I. Transmiterea parametrilor în proceduri

Tipul implicit **IN**

Daca nu este specificat nici un tip parametrului, atunci implicit este **IN**.

Sintaxa

CREATE PROCEDURE

procedure(param [mode] datatype)

...

Exemplu:

CREATE OR REPLACE PROCEDURE

raise_sal

**(p_id IN emp.empno%TYPE, p_percent IN
NUMBER)**

IS

BEGIN

UPDATE emp

SET sal = sal * (1 + p_percent/100)

WHERE empno = p_id;

END raise_sal;

Parametrii **IN** sunt doar date de intrare pentru
procedura.

Ei nu pot fi modificati in interiorul procedurii.

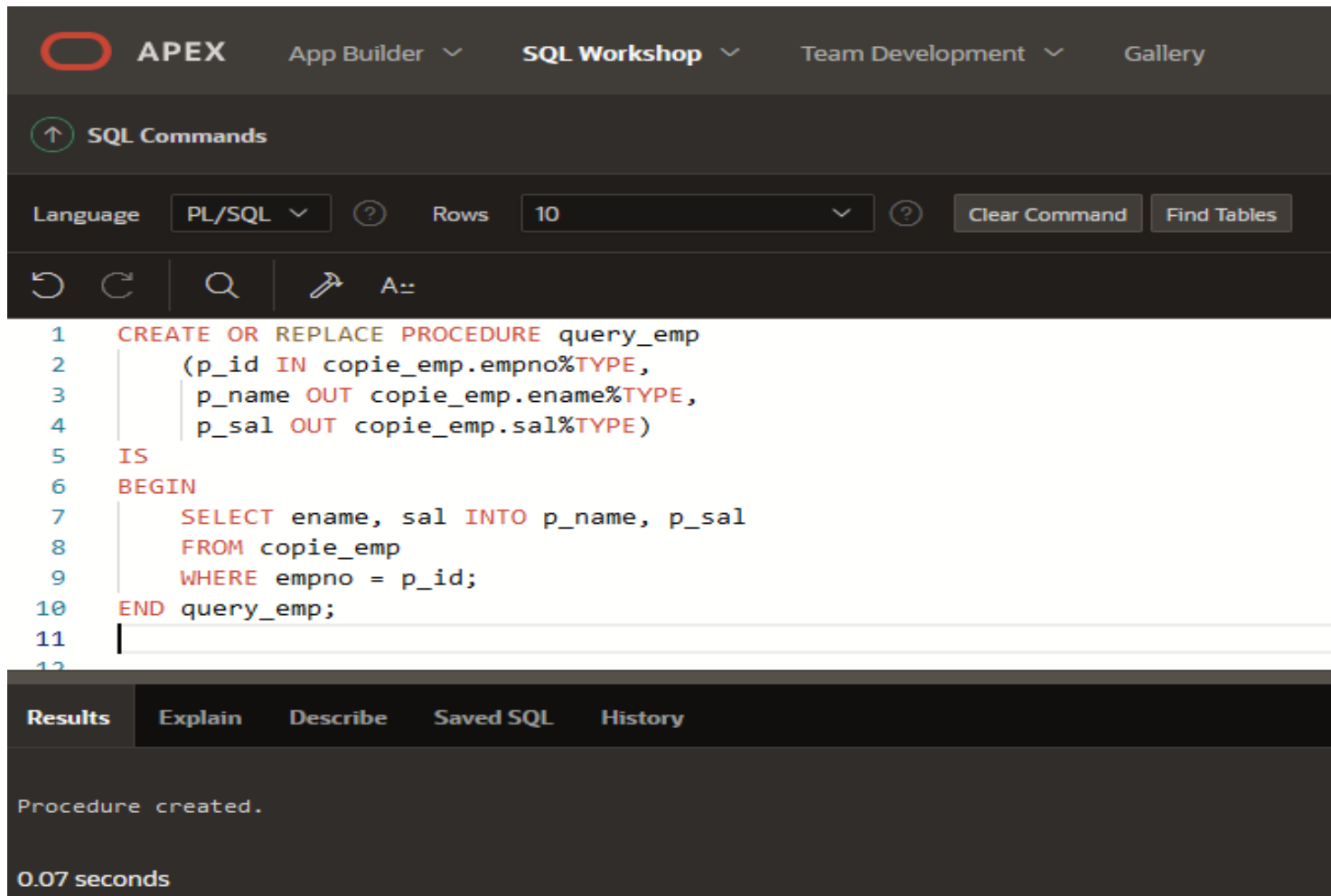
I. Transmiterea parametrilor în proceduri

*Exemplu – folosirea parametrilor **OUT***

```
CREATE OR REPLACE PROCEDURE query_emp  
(p_id IN emp.empno%TYPE,  
  p_name OUT emp.ename%TYPE,  
  p_sal OUT emp.sal%TYPE)  
IS  
BEGIN  
  SELECT ename, sal INTO p_name, p_sal  
  FROM emp  
  WHERE empno = p_id;  
END query_emp;
```

I. Transmiterea parametrilor în proceduri

Exemplu – folosirea parametrilor **OUT**



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing the 'Language' set to 'PL/SQL' and 'Rows' set to '10'. The main area contains the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE query_emp
2   (p_id IN copie_emp.empno%TYPE,
3    p_name OUT copie_emp.ename%TYPE,
4    p_sal OUT copie_emp.sal%TYPE)
5 IS
6 BEGIN
7   SELECT ename, sal INTO p_name, p_sal
8   FROM copie_emp
9   WHERE empno = p_id;
10 END query_emp;
```

Below the code editor, the 'Results' tab is selected, displaying the message 'Procedure created.' and the execution time '0.07 seconds'.

I. Transmiterea parametrilor în proceduri

DECLARE

a_emp_name emp.ename%TYPE;

a_emp_sal emp.sal%TYPE;

BEGIN

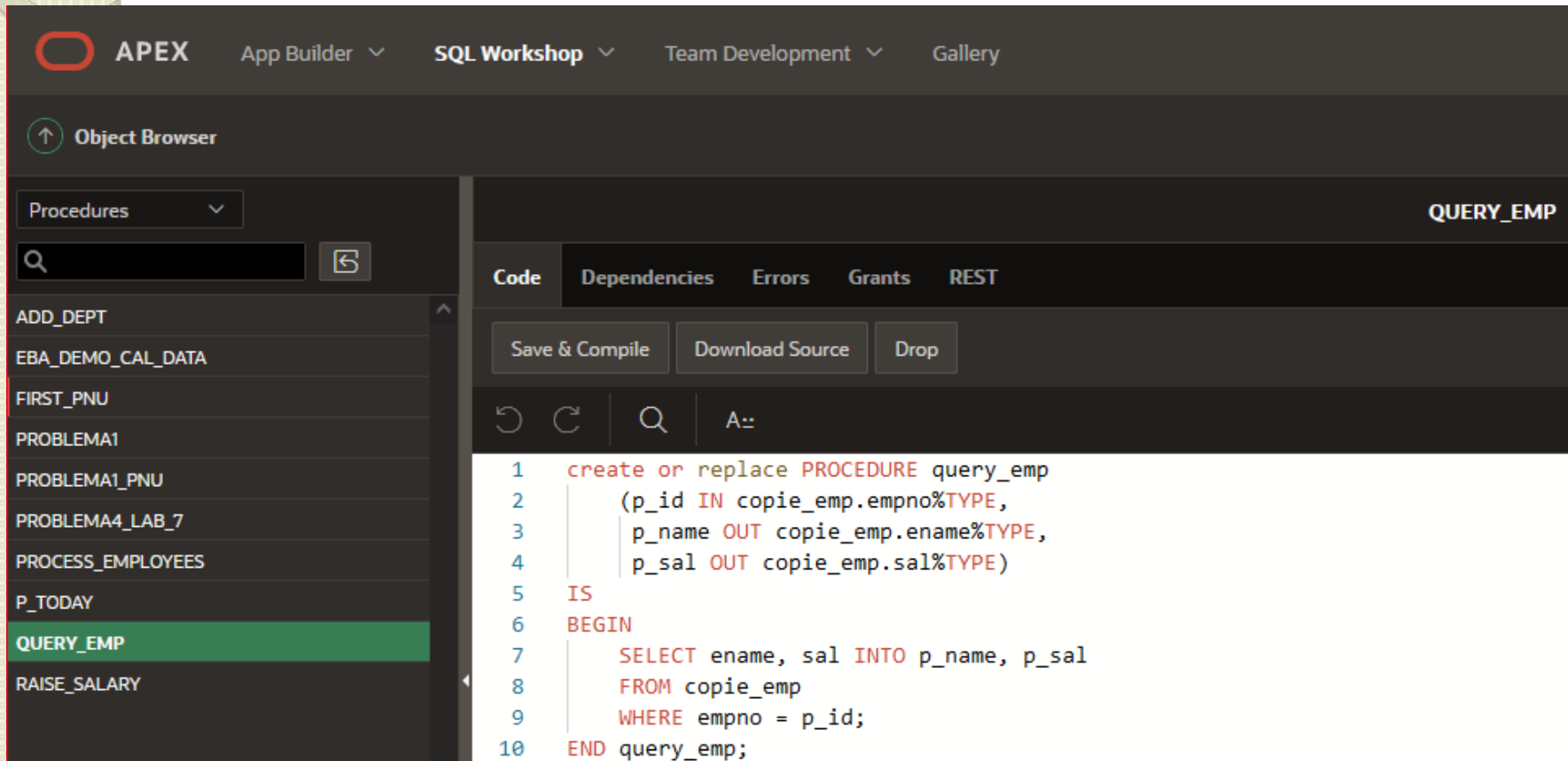
**query_emp(7566, a_emp_name,
a_emp_sal);**

**DBMS_OUTPUT.PUT_LINE('Name: ' ||
a_emp_name);**

**DBMS_OUTPUT.PUT_LINE('sal: ' ||
a_emp_sal);**

END;

I. Transmiterea parametrilor în proceduri

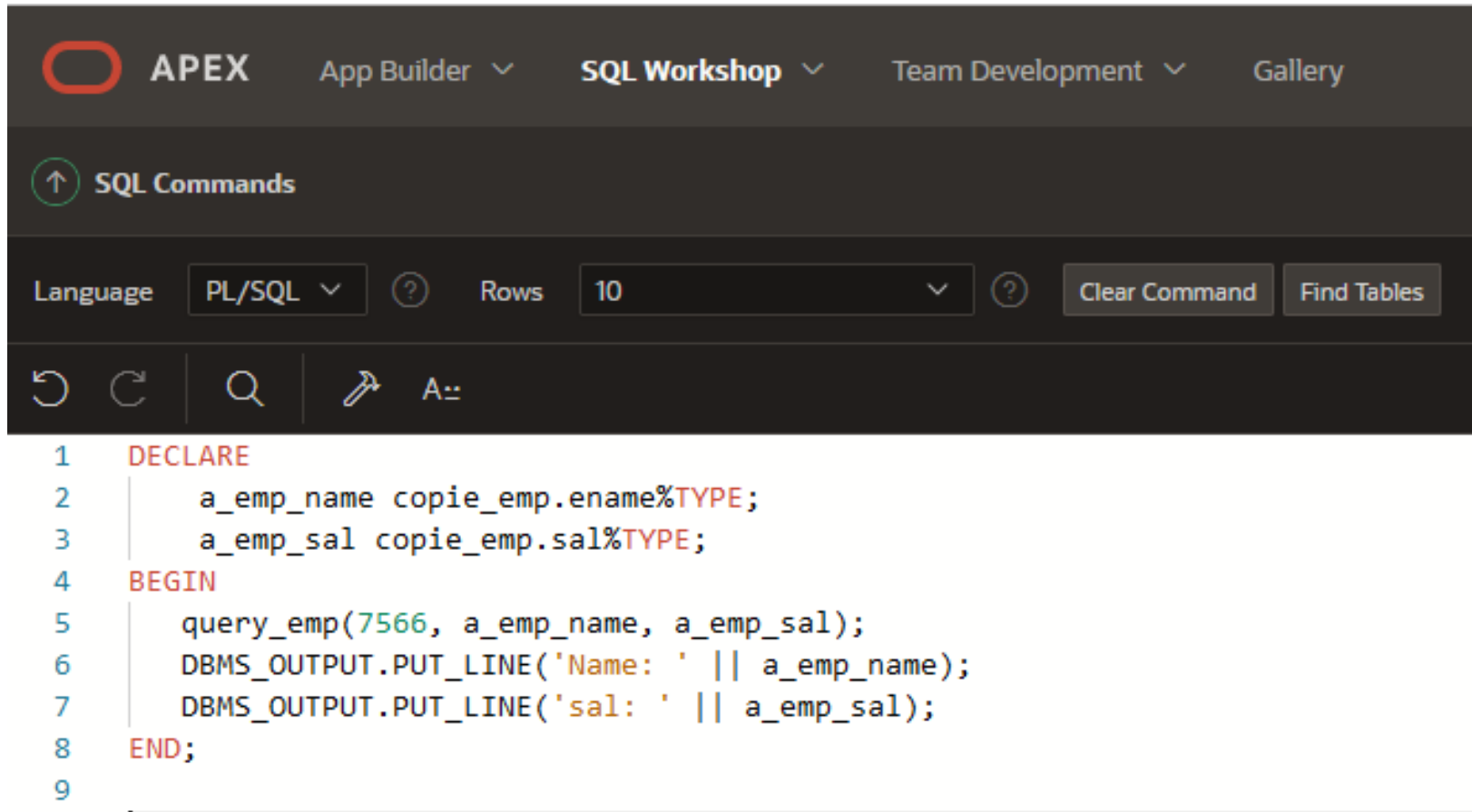


The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes the APEX logo and menu items: App Builder, SQL Workshop, Team Development, and Gallery. On the left, the Object Browser shows a tree view of procedures, with 'QUERY_EMP' selected and highlighted in green. The main workspace is titled 'QUERY_EMP' and contains a code editor with the following SQL code:

```
1 create or replace PROCEDURE query_emp
2   (p_id IN copie_emp.empno%TYPE,
3   p_name OUT copie_emp.ename%TYPE,
4   p_sal OUT copie_emp.sal%TYPE)
5   IS
6   BEGIN
7     SELECT ename, sal INTO p_name, p_sal
8     FROM copie_emp
9     WHERE empno = p_id;
10  END query_emp;
```

Below the code editor, there are buttons for 'Save & Compile', 'Download Source', and 'Drop'. At the bottom of the editor, there are navigation icons for back, forward, search, and a command prompt icon.

I. Transmiterea parametrilor în proceduri



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is visible, with 'Language' set to 'PL/SQL' and 'Rows' set to '10'. The main area displays the following PL/SQL code:

```
1 DECLARE
2     a_emp_name copie_emp.ename%TYPE;
3     a_emp_sal copie_emp.sal%TYPE;
4 BEGIN
5     query_emp(7566, a_emp_name, a_emp_sal);
6     DBMS_OUTPUT.PUT_LINE('Name: ' || a_emp_name);
7     DBMS_OUTPUT.PUT_LINE('sal: ' || a_emp_sal);
8 END;
9
```

I. Transmiterea parametrilor în proceduri

QUERY_EMP

Code Dependencies Errors Grants

Save & Compile

Find & Replace

Undo

Redo

Download Source

Drop

PL/SQL code successfully compiled (15:22:58)

```

1 create or replace PROCEDURE query_emp (p_id IN emp.empno%TYPE, p_name OUT emp.ename%TYPE, p_sal OUT emp.sal%TYPE) IS
2 BEGIN
3     SELECT ename, sal INTO p_name, p_sal
4     FROM emp
5     WHERE empno = p_id;
6 END query_emp;

```

ORACLE Application Express

Home

Application Builder ▾

SQL Workshop ▾

Team Development ▾

Administration ▾



SQL Workshop

SQL Commands

Rows

10



Save

Run

```

DECLARE
a_emp_name emp.ename%TYPE;
a_emp_sal emp.sal%TYPE;
BEGIN
    query_emp(7566, a_emp_name, a_emp_sal);
    DBMS_OUTPUT.PUT_LINE('Name: ' || a_emp_name);
    DBMS_OUTPUT.PUT_LINE('sal: ' || a_emp_sal);
END;

```

- *Procedura a fost creata pentru a extrage informatii despre un anumit candidat.*
- Procedura primeste valoarea **7566** pentru id-ul angajatului si returneaza numele si salariul angajatului cu id-ul **7566** in cei doi parametri **OUT**.
- Procedura **query_emp** are trei parametri formali.
- Doi dintre ei sunt de tip **OUT** care returneaza valori programului appellant.
- Procedura primeste valoarea id-ului angajatului prin intermediul parametrului **p_id**.
- Variabilele **a_emp_name** si **a_emp_sal** sunt completate cu informatii preluate din interogare in cei doi parametri **OUT** corespunzatori.

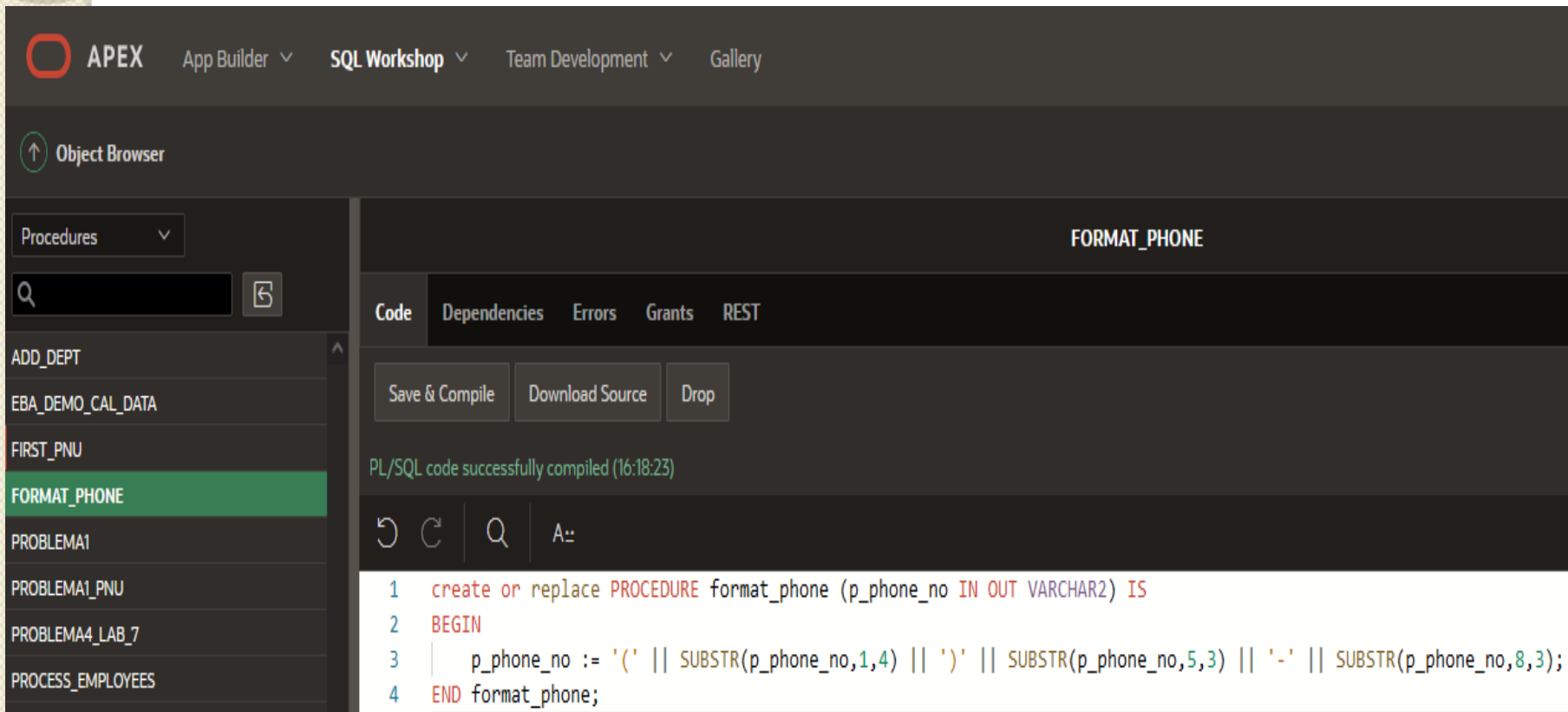
I. Transmiterea parametrilor în proceduri

*Exemplu – folosirea parametrilor **IN OUT***

```
create or replace PROCEDURE  
  format_phone (p_phone_no IN OUT  
  VARCHAR2) IS  
BEGIN  
  p_phone_no := '(' ||  
  SUBSTR(p_phone_no,1,4) || ')' ||  
  SUBSTR(p_phone_no,5,3) || '-' ||  
  SUBSTR(p_phone_no,8,3);  
END format_phone;
```


I. Transmiterea parametrilor în proceduri

Exemplu – folosirea parametrilor **IN OUT**



The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'Object Browser' on the left shows a list of procedures, with 'FORMAT_PHONE' selected. The main workspace shows the 'Code' tab for the 'FORMAT_PHONE' procedure. The code is as follows:

```
1 create or replace PROCEDURE format_phone (p_phone_no IN OUT VARCHAR2) IS
2 BEGIN
3     p_phone_no := '(' || SUBSTR(p_phone_no,1,4) || ')' || SUBSTR(p_phone_no,5,3) || '-' || SUBSTR(p_phone_no,8,3);
4 END format_phone;
```

Below the code editor, a message states 'PL/SQL code successfully compiled (16:18:23)'. The interface also shows buttons for 'Save & Compile', 'Download Source', and 'Drop'.

I. Transmiterea parametrilor în proceduri

- Folosind un parametru **IN OUT**, puteți transmite o valoare procedurii care poate fi modificată de procedura.
- Valoarea parametrului actual primită la apel poate fi returnată în una dintre următoarele variante:
 1. valoarea inițială nemodificată
 2. o nouă valoare din procedura

In exemplul anterior:

- inainte de apel variabila ***p_phone_no*** are valoarea **'0720123456'**
- dupa apel variabila ***p_phone_no*** are valoarea **'(0721)123-456'**

Urmatorul cod creeaza un bloc anonim care declara ***a_phone_no***, ii atribuie un numar de telefon neformatat si il transmite ca parametru actual procedurii ***FORMAT_PHONE***.

Procedura este executata si returneaza un sir de caractere modificat in variabila ***a_phone_no*** care este apoi afisata.

I. Transmiterea parametrilor în proceduri

```
DECLARE
```

```
  a_phone_no VARCHAR2(13);
```

```
BEGIN
```

```
  a_phone_no := '0721123456';
```

```
  format_phone (a_phone_no);
```

```
  DBMS_OUTPUT.PUT_LINE('The  
formatted phone number is: '|  
a_phone_no);
```

```
END;
```

I. Transmiterea parametrilor în proceduri

The screenshot displays the Oracle SQL Developer interface. At the top, the window title is "FORMAT_PHONE". Below the title bar, there are tabs for "Code", "Dependencies", "Errors", "Grants", and "REST". A toolbar contains buttons for "Save & Compile", "Download Source", and "Drop". A status bar indicates "PL/SQL code successfully compiled (16:18:23)".

The main editor shows the following PL/SQL code:

```
1 create or replace PROCEDURE format_phone (p_phone_no IN OUT VARCHAR2) IS
2 BEGIN
3     p_phone_no := '(' || SUBSTR(p_phone_no,1,4) || ')' || SUBSTR(p_phone_no,5,3) || '-' || SUBSTR(p_phone_no,8,3);
4 END format_phone;
```

Below the code editor, there is a toolbar with "Language" set to "PL/SQL", "Rows" set to "10", and buttons for "Clear Command" and "Find Tables". The execution area shows the following code:

```
1 DECLARE
2     p_phone_no VARCHAR2(20);
3 BEGIN
4     p_phone_no := '0720123456';
5     format_phone(p_phone_no);
6     DBMS_OUTPUT.PUT_LINE(p_phone_no);
7 END;
```

The "Results" tab is active, showing the output of the procedure:

```
(0720)123-456
Statement processed.
0.01 seconds
```

Sintaxa pentru transmiterea parametrilor

Sunt trei modalitati de transmitere a parametrilor de la mediul apelant:

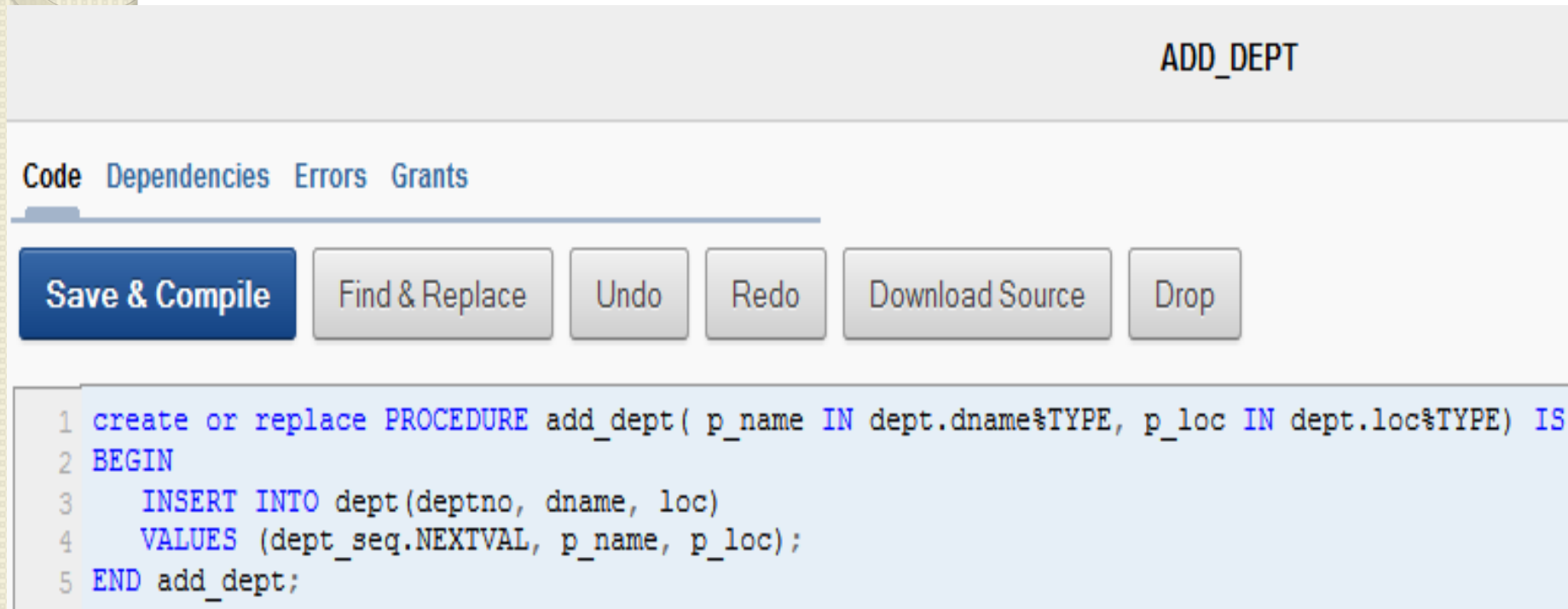
1. **pozitional** – listarea parametrilor actuali in aceeași ordine ca și cei formali
2. **prin denumire** – listarea parametrilor actuali într-o ordine arbitrara și folosirea operatorului de asociere ($=>$) pentru a asocia un parametru formal denumit cu parametrul actual corespunzator
3. **combinat** – listarea catorva parametri pozitionali (fara operator special) și altii prin denumire (cu operatorul $=>$)

I. Transmiterea parametrilor în proceduri

Exemple:

```
CREATE OR REPLACE PROCEDURE add_dept  
  (p_name IN dept.dname%TYPE,  
   p_loc IN dept.loc%TYPE)  
IS  
BEGIN  
  INSERT INTO dept(deptno, dname, loc)  
  VALUES (dept_seq.NEXTVAL, p_name, p_loc);  
END add_dept;
```

I. Transmiterea parametrilor în proceduri



The screenshot displays the Oracle SQL Developer interface for a procedure named 'ADD_DEPT'. The interface includes a title bar with the procedure name, a menu bar with 'Code', 'Dependencies', 'Errors', and 'Grants', and a toolbar with buttons for 'Save & Compile', 'Find & Replace', 'Undo', 'Redo', 'Download Source', and 'Drop'. The main area contains the following PL/SQL code:

```
1 create or replace PROCEDURE add_dept( p_name IN dept.dname%TYPE, p_loc IN dept.loc%TYPE) IS
2 BEGIN
3     INSERT INTO dept(deptno, dname, loc)
4     VALUES (dept_seq.NEXTVAL, p_name, p_loc);
5 END add_dept;
```


I. Transmiterea parametrilor în proceduri

1. Transmiterea prin notatie pozitionala
add_dept ('EDUCATION', 1400);

2. Transmiterea prin denumire
*add_dept (p_loc=>1400,
p_name=>'EDUCATION');*

3. Transmiterea prin notatie combinata
add_dept ('EDUCATION', p_loc=>1400);

I. Transmiterea parametrilor în proceduri

Urmatorul apel se va executa cu succes?

add_dept (p_loc => 1400, 'EDUCATION');

Raspuns:

Nu – deoarece atunci cand se foloseste notatia combinata, *parametrii notati pozitional trebuie sa fie scrisi inaintea celor transmisi prin denumire.*

I. Transmiterea parametrilor în proceduri

Urmatorul apel se va executa cu succes?

add_dept ('EDUCATION');

Raspuns:

Nu - *trebuie furnizata o valoare pentru fiecare parametru in afara de cazul cand parametrului formal ii este atribuita o valoare implicita.*

I. Transmiterea parametrilor în proceduri

Folosirea optiunii **DEFAULT** pentru parametrii **IN**

- Puteti atribui valori implicite parametrilor formali **IN** - exemplu

```
CREATE OR REPLACE PROCEDURE add_dept (  
p_name dept.dname%TYPE:='Unknown',  
p_loc dept.loc%TYPE DEFAULT 1400)  
IS  
BEGIN  
  INSERT INTO dept (...)  
  VALUES (dept_seq.NEXTVAL, p_name, p_loc);  
END add_dept;
```

I. Transmiterea parametrilor în proceduri

Codul prezinta doua modalitati de atribuire a unei valori implicite unui parametru **IN**.
Cele doua modalitati prezentate folosesc:

1. **operatorul de atribuire (:=)** – pentru parametrul ***p_name***
2. **optiunea DEFAULT** – pentru operatorul ***p_loc***

I. Transmiterea parametrilor în proceduri

Prezentam trei modalitati de apelare a procedurii ***add_dept***

1. **Atribuirea de valori implicite pentru fiecare parametru**
2. **Combinarea notatiilor pozitionale si denumite pentru atribuirea de valori**
3. **Folosirea de valori implicite pentru parametrii cu nume si de valori transmise pentru ceilalti parametri**

I. Transmiterea parametrilor în proceduri

1. **add_dept;**
2. **add_dept ('ADVERTISING', p_loc => 1400);**
3. **add_dept (p_loc => 1400);**

Reguli de folosire a optiunii *DEFAULT* pentru parametri

- Nu puteti folosi valori implicite parametrilor **OUT** si **IN OUT** in antet, dar acest lucru se poate realiza in corpul procedurii.
- De obicei, puteti folosi denumirile pentru a suprascrie valorile implicite ale parametrilor formali. Totusi, nu puteti sari peste operatia de transmitere a unui parametru actual daca nu este nici o valoare implicita pentru parametrul formal.
- Un parametru care mosteneste o valoare implicita este diferit de **NULL**.

Lucrul cu erorile de parametri in timpul rularii

- *Toti parametrii pozitionali trebuie sa preceada parametrii denumiti in apelul unui subprogram.*
- Altfel este afisat un mesaj de eroare cum se arata in urmatorul exemplu:

BEGIN

add_dept (name =>'new dept', 'new location');

END;

Se genereaza urmatorul mesaj de eroare:

```
ORA-06550: line 2, column 2:  
PLS-00306: wrong number or types of arguments in call to 'ADD_DEPT'  
ORA-06550: line 2, column 2:  
PL/SQL: Statement ignored
```

```
1. BEGIN  
2.     add_dept (name =>'new dept', 'new location');  
3. END;
```

Subprograme locale

- ° Atunci cand o procedura apeleaza alta procedura, in mod normal le cream separat.

```
CREATE OR REPLACE PROCEDURE subproc
```

```
...
```

```
END subproc;
```

```
CREATE OR REPLACE PROCEDURE mainproc
```

```
...
```

```
IS BEGIN
```

```
...
```

```
subproc (...);
```

```
...
```

```
END mainproc;
```

Dar se pot crea si impreuna ca o singura procedura.

CREATE OR REPLACE PROCEDURE mainproc

...

IS

PROCEDURE subproc (...) IS BEGIN

...

END subproc;

BEGIN

...

subproc (...);

...

END mainproc;

- Aici tot codul este intr-un singur loc si este mai usor de citit si de intretinut.
- Domeniul unui subprogram imbricat este limitat la procedura in care este definit;
- SUBPROC poate fi apelat din MAINPROC dar nu si din alt subprogram sau din program.



Întrebări?