Tehnici de programare cu baze de date

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

https://www.runceanu.ro/adrian/activitate-didactica/

Curs 9

Proceduri în PL/SQL
(partea a II-a)

Cuprins

Proceduri în PL/SQL

1. Transmiterea parametrilor în proceduri

1.Transmiterea parametrilor în proceduri

- Pentru ca procedurile sa fie mai flexibile este important ca diverse date sa fie fie transmise procedurii prin parametrii de intrare.

- Rezultatele determinate pot fi returnate prin folosirea parametrilor de tip ***OUT*** sau ***IN OUT***.

1.Transmiterea parametrilor în proceduri

Tipuri de parametri procedurali

Tipurile parametrilor sunt specificate in declararea parametrilor formali, dupa numele parametrului si inainte de tipul sau de date.

1.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.*

1.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.

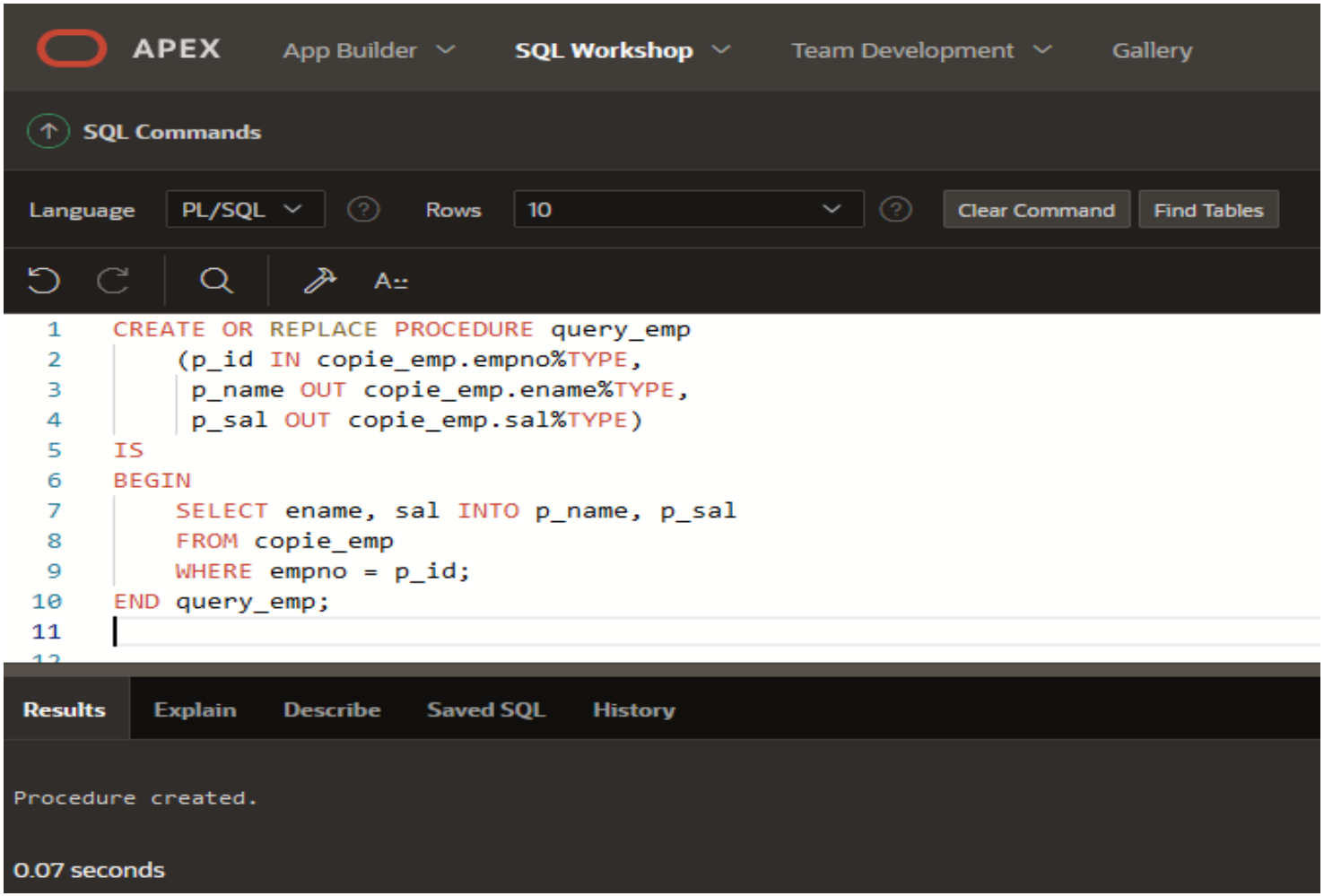
1.Transmiterea parametrilor în proceduri

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;
```

1.Transmiterea parametrilor în proceduri

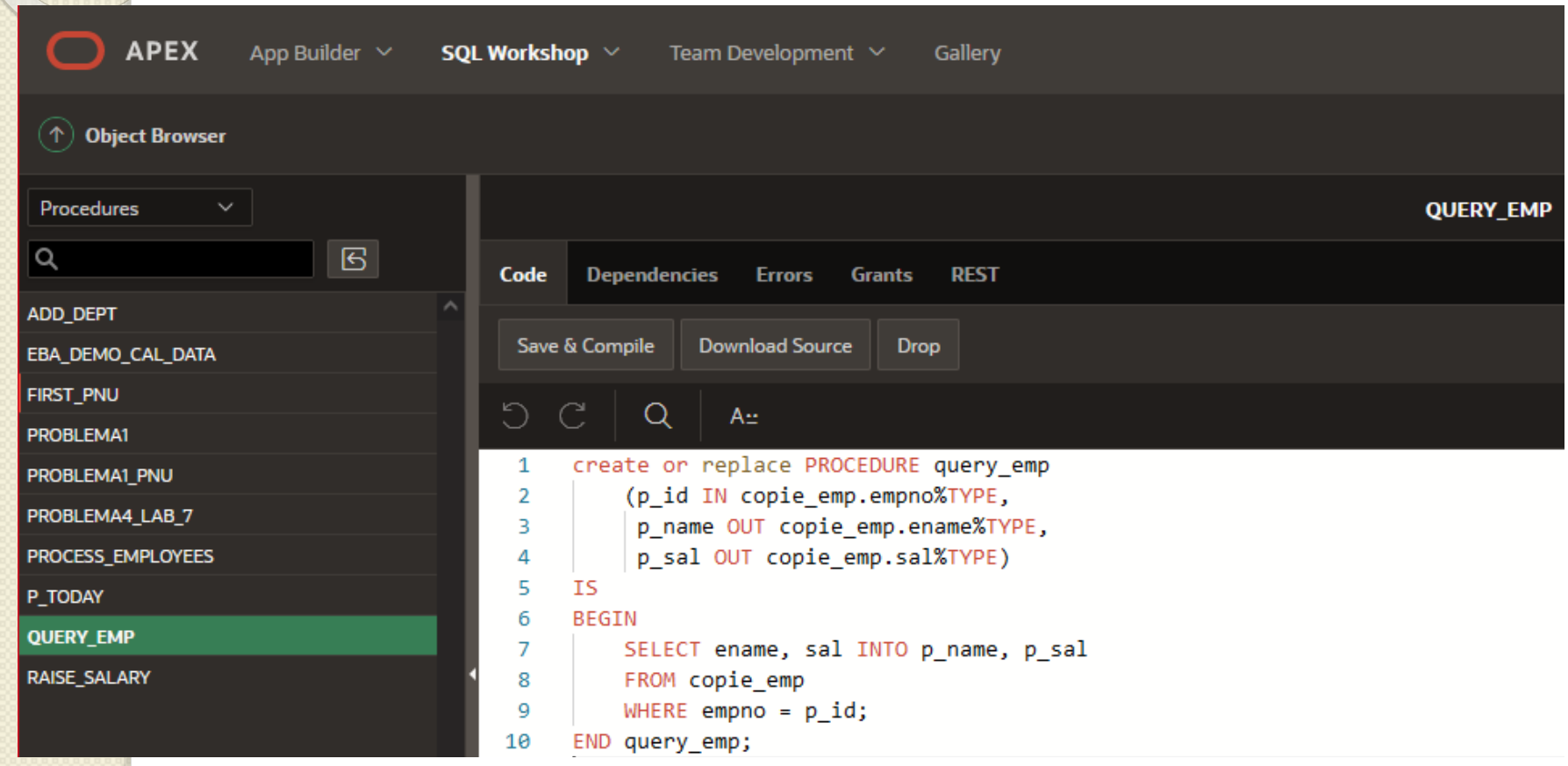
Exemplu – folosirea parametrilor OUT



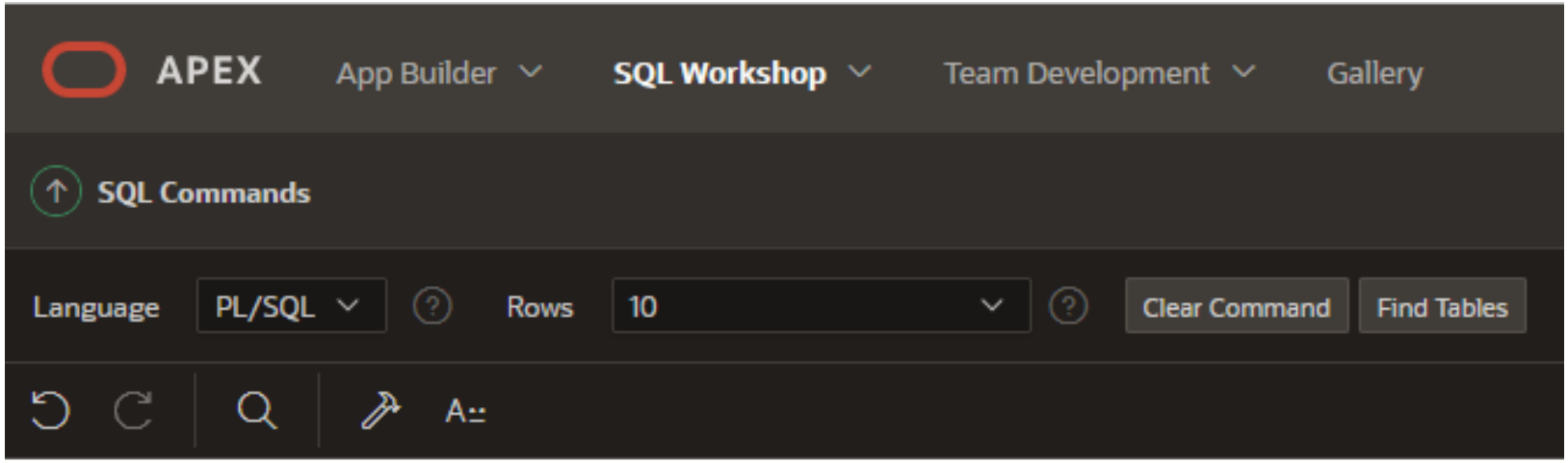
1.Transmiterea parametrilor în proceduri

```
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;
```

1.Transmiterea parametrilor în proceduri



1.Transmiterea parametrilor în proceduri




```
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
```

1.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.

1.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;
```

1.Transmiterea parametrilor în proceduri

Exemplu – folosirea parametrilor IN OUT

1.Transmiterea parametrilor în proceduri

- *Folosind un parametru* IN OUT, *puteti transmite o valoare procedurii care poate fi modificata de procedura.*

- Valoarea parametrului actual primita la apel poate fi returnata in una dintre urmatoarele variante:
  1. valoarea initiala nemodificata
  2. o noua 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.

1.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;
```

1.Transmiterea parametrilor în proceduri

FORMAT_PHONE

Code | Dependencies | Errors | Grants | REST

Save & Compile | Download Source | Drop

PL/SQL code successfully compiled (16:18:23)

```
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;
```

Development | Gallery

Language PL/SQL | Rows 10 | Clear Command | Find Tables

```
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;
8
```

Results | Explain | Describe | Saved SQL | History

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

1.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;
```

1.Transmiterea parametrilor în proceduri

ADD_DEPT

Code | Dependencies | Errors | Grants

Download | Save and Compile | Drop | Refresh

```
1  create or replace PROCEDURE add_dept
2              (p_cod dept.deptno%TYPE,p_nume dept.dname%TYPE,p_location dept.loc%TYPE)
3  IS
4  BEGIN
5      INSERT INTO dept
6      VALUES (p_cod,p_nume,p_location);
7  END;
```

1.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);

1.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*.

1.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.*

1.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;
```

1.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***

1.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

1.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?