

Cuprins

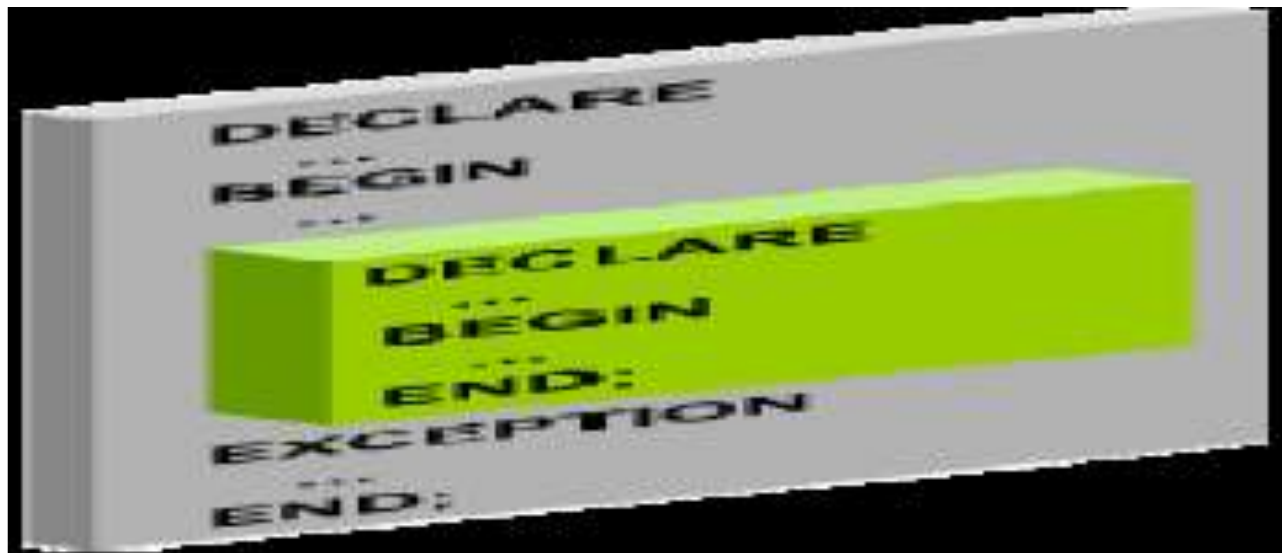
Proceduri in PL/SQL

- 1. Exceptii. Domeniul variabilelor -
recapitulare**
- 2. Proceduri și funcții**
- 3. Folosirea parametrilor în proceduri**

RECAPITULARE

- O exceptie denumita este un fel de variabila PL/SQL.
- Pentru manipularea corecta a exceptiilor trebuie sa intelegeti:
 - domeniul
 - si vizibilitatea variabilelor exceptiilor
- Acest lucru este deosebit de important in cazul blocurilor imbricate.

- **Blocuri imbricate** – exista un bloc exterior si un bloc interior.
- Blocurile pot fi imbricate pe oricate nivele, nu exista nici o restrictie in acest sens.



Exemplu:

Un bloc exterior – parinte (reprezentat in **albastru**) si un bloc interior – copil (reprezentat in **rosu**).

Variabila **v_outer_variable** este declarata in blocul exterior si variabila **v_inner_variable** este declarata in blocul interior.

DECLARE

v_outer_variable VARCHAR2(20):='GLOBAL VARIABLE';

BEGIN

DECLARE

v_inner_variable VARCHAR2(20):='LOCAL VARIABLE';

BEGIN

DBMS_OUTPUT.PUT_LINE(v_inner_variable);

DBMS_OUTPUT.PUT_LINE(v_outer_variable);

END;

DBMS_OUTPUT.PUT_LINE(v_outer_variable);

END;

ORACLE® Application Express

Home > SQL Workshop > SQL Commands

Autocommit Display 10  

```
DECLARE
v_outer_variable VARCHAR2(20):='GLOBAL VARIABLE';
BEGIN
    DECLARE
v_inner_variable VARCHAR2(20):='LOCAL VARIABLE';
    BEGIN
        DBMS_OUTPUT.PUT_LINE(v_inner_variable);
        DBMS_OUTPUT.PUT_LINE(v_outer_variable);
    END;
DBMS_OUTPUT.PUT_LINE(v_outer_variable);
END;
```

Results Explain Describe Saved SQL History

```
LOCAL VARIABLE
GLOBAL VARIABLE
GLOBAL VARIABLE
```

Statement processed.

0.28 seconds

Domeniul de aplicare a unei variabile este blocul sau blocurile in care variabila este accesibila.

In PL/SQL domeniul unei variabile este blocul in care este declarata si in toate blocurile imbricate in interiorul blocului declarativ.
Care este domeniul celor doua variabile declarate in exemplul anterior?
Fie urmatorul *exemplu*:

DECLARE

v_father_name VARCHAR2(20):='Patrick';

v_date_of_birth DATE:='20-Apr-1972';

BEGIN

DECLARE

v_child_name VARCHAR2(20):='Mike';

BEGIN

**DBMS_OUTPUT.PUT_LINE('Father"s Name:
'||v_father_name);**

**DBMS_OUTPUT.PUT_LINE('Date of Birth:
'||v_date_of_birth);**

**DBMS_OUTPUT.PUT_LINE('Child"s Name:
'||v_child_name);**

END;

DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);

END;

ORACLE Application Express

Home > SQL Workshop > SQL Commands

Autocommit Display 10  

```
DECLARE
v_father_name VARCHAR2(20):='Patrick';
v_date_of_birth DATE:='20-Apr-1972';
BEGIN
    DECLARE
        v_child_name VARCHAR2(20):='Mike';
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Father's Name: '||v_father_name);
        DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
        DBMS_OUTPUT.PUT_LINE('Child's Name: '||v_child_name);
    END;
DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
END;
```

Results Explain Describe Saved SQL History

```
Father's Name: Patrick
Date of Birth: 20-04-1972
Child's Name: Mike
Date of Birth: 20-04-1972
```

Statement processed.

0.01 seconds.

De ce urmatorul cod nu va functiona corect?

BEGIN

DECLARE

**CURSOR emp_curs IS SELECT * FROM emp;
v_emp_rec emp_curs%ROWTYPE;**

BEGIN

OPEN emp_curs;

LOOP

FETCH emp_curs INTO v_emp_rec;

EXIT WHEN emp_curs%NOTFOUND;

DBMS_OUTPUT.PUT_LINE(v_emp_rec.ename);

END LOOP;

END;

CLOSE emp_curs;

END;

Rows

10

Save

Run

```
BEGIN
  DECLARE
    CURSOR emp_curs IS SELECT * FROM emp;
    v_emp_rec emp_curs%ROWTYPE;
  BEGIN
    OPEN emp_curs;
    LOOP
      FETCH emp_curs INTO v_emp_rec;
      EXIT WHEN emp_curs%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_emp_rec.ename);
    END LOOP;
  END;
CLOSE emp_curs;
END;
```

Cursorul este vizibil doar in
blocul unde a fost definit

```
ORA-06550: line 13, column 7:
PLS-00201: identifier 'EMP_CURS' must be declared
ORA-06550: line 13, column 1:
PL/SQL: SQL Statement ignored
```

```
1. BEGIN
2.     DECLARE
3.         CURSOR emp_curs IS SELECT * FROM emp;
```

Urmatorul cod va functiona corect? Justificati.

```
DECLARE  
CURSOR emp_curs IS SELECT * FROM emp;  
BEGIN  
  OPEN emp_curs;  
  DECLARE  
    v_emp_rec emp_curs%ROWTYPE;  
  BEGIN  
    LOOP  
      FETCH emp_curs INTO v_emp_rec;  
      EXIT WHEN emp_curs%NOTFOUND;  
      DBMS_OUTPUT.PUT_LINE(v_emp_rec.ename);  
    END LOOP;  
  END;  
  CLOSE emp_curs;  
END;
```

Rows

10

Save

Run

```
DECLARE
  CURSOR emp_curs IS SELECT * FROM emp;
BEGIN
  OPEN emp_curs;
  DECLARE
    v_emp_rec emp_curs%ROWTYPE;
  BEGIN
    LOOP
      FETCH emp_curs INTO v_emp_rec;
      EXIT WHEN emp_curs%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_emp_rec.ename);
    END LOOP;
  END;
  CLOSE emp_curs;
END;
```

Results Explain Describe Saved SQL History

KING
BLAKE
CLARK
JONES
SCOTT
FORD
SMITH
ALLEN
WARD
MARTIN
TURNER
ADAMS
JAMES
MILLER
Vasilica

Statement processed.

0.02 seconds

Cum rezolva PL/SQL denumirile de variabile – recapitulare

Atunci cand referiti numele unei variabile intr-un bloc,
PL/SQL cauta mai intai sa vada daca o variabila cu acel nume a
fost declarata in blocul respectiv (variabila locala).
Daca nu o gaseste, PL/SQL cauta in blocul exterior etc.

```
DECLARE           -- outer block
  v_outervar VARCHAR2(20);
BEGIN
  DECLARE         -- middle-level block
    v_middlevar VARCHAR2(20);
  BEGIN          -- innermost block
    v_outervar := 'Joachim';
    v_middlevar := 'Chang';
  END;
END;
END;
```

Manipularea exceptiilor in blocurile imbricate

Puteti trata exceptia prin:

- Manipularea ei in blocul in care apare
- Propagarea ei in mediul apelant (care poate fi un bloc de un nivel superior)

Propagarea exceptiilor catre blocul exterior

- Daca se produce o exceptie in sectiunea executabila a blocului interior si nu este nici un manipulator de exceptie corespunzator, blocul PL/SQL se incheie cu esec si exceptia este propagata in blocul exterior.
- In urmatorul exemplu apare o exceptie in timpul executiei blocului interior.
- Sectiunea de exceptie a blocului interior nu reuseste sa trateze exceptia.
- Blocul interior se incheie fara succes si PL/SQL propaga exceptia catre blocul exterior.
- Sectiunea de exceptie a blocului exterior manipuleaza cu success exceptia.

```
DECLARE      -- outer block
  e_no_rows EXCEPTION;
BEGIN
  BEGIN      -- inner block
  IF ... THEN RAISE e_no_rows;      -- exception
  occurs here
  ...
  END;
  ...      -- Remaining code in outer block's
  executable
  ...      -- section is skipped
  EXCEPTION
  WHEN e_no_rows THEN -- outer block handles
  the exception

  ...
END;
```

- Daca **PL/SQL** produce o exceptie si blocul curent nu are un manipulator pentru acea exceptie, acea exceptie se propaga succesiv blocurilor exterioare pana cand este gasit un manipulator.
- Atunci cand exceptia se propaga catre blocul exterior, operatiile executabile ramase sunt ignorate, nu se mai executa.
- Un avantaj al acestui lucru este ca puteti sa adaugati instructiuni care necesita propriile manipulari de erori in propriile blocuri, in timp ce pastram manipularile exceptiilor mai generale (de exemplu **WHEN OTHERS**) pentru blocul apelant.

Exemplu – propagarea exceptiilor predefinite dintr-un sub-bloc

DECLARE

v_last_name emp.ename%TYPE;

BEGIN

BEGIN

**SELECT ename INTO v_last_name
FROM emp**

WHERE empno = 999;

DBMS_OUTPUT.PUT_LINE('Message 1');

EXCEPTION

WHEN TOO_MANY_ROWS THEN

DBMS_OUTPUT.PUT_LINE('Message 2');

END;

DBMS_OUTPUT.PUT_LINE('Message 3');

EXCEPTION

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Message 4');

END;

Employee_id 999 nu exista. Ce se afiseaza atunci cand este executat codul anterior?

Rows

10

Save

Run

```
DECLARE
    v_last_name emp.ename%TYPE;
BEGIN
    BEGIN
        SELECT ename INTO v_last_name
        FROM emp WHERE empno = 999;
        DBMS_OUTPUT.PUT_LINE('Message 1');
    EXCEPTION
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Message 2');
    END;
    DBMS_OUTPUT.PUT_LINE('Message 3');
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Message 4');
END;
```

Results Explain Describe Saved SQL History

Message 4

Statement processed.

0.01 seconds

Domeniul de aplicare al numelor de exceptii

- Exceptiile predefinite ale serverului Oracle cum ar fi ***NO_DATA_FOUND, TOO_MANY_ROWS*** si ***OTHERS*** nu sunt declarate de catre programator. Pot fi produse in orice bloc si manipulate in orice bloc.
- Exceptiile denumite de utilizator sunt declarate de programator ca variabile de tipul ***EXCEPTION***. Ele respecta aceleasi reguli ca si celelalte variabile.
- Prin urmare, o exceptie definita de utilizator declarata intr-un bloc interior nu poate fi referita in sectiunea de exceptie a unui bloc exterior.

Cuprins

- 1. Exceptii. Domeniul variabilelor - recapitulare**
- 2. Proceduri și funcții**
- 3. Folosirea parametrilor în proceduri**

2. Proceduri

Crearea procedurilor

- Am studiat cum sa scriem si sa executam ***blocuri PL/SQL anonime***.
- Blocurile anonime sunt scrise ca parte a programului aplicatie.
- Acum vom studia cum sa creem, sa executam si sa administram ***subprogramele PL/SQL***.
- Acestea sunt stocate in baza de date, oferind multe beneficii cum ar fi o mai buna securitate si rapiditate.

2. Proceduri

Sunt doua tipuri de ***subprograme PL/SQL***:

1. ***Proceduri***
2. ***Functii***

Diferenta dintre blocuri anonime si subprograme

- **Blocurile anonime** – sunt *blocuri executabile PL/SQL fara nume*.
- Deoarece sunt nedenumite, ele nu pot fi nici reutilizate sau stocate in baza de date pentru o folosire ulterioara.

Diferenta dintre blocuri anonime si subprograme

- **Subprogramele** – procedurile si functiile sunt *blocuri PL/SQL denumite*.
- Sunt cunoscute sub denumirea de subprograme.
- Aceste subprograme sunt compilate si stocate in baza de date.
- Structura blocului unui subprogram este asemanatoare cu structura unui bloc anonim.
- In timp ce subprogramele pot fi partajate in mod explicit, implicit este de a le face **private** schemei proprii.

Mai tarziu *subprogramele devin blocuri in constructia pachetelor si triggerelor (declansatorilor)*.

Anonymous Blocks

DECLARE (Optional)

Variables, cursors, etc.;

BEGIN (Mandatory)

SQL and PL/SQL statements;

EXCEPTION (Optional)

WHEN exception-handling actions;

END; (Mandatory)

Subprograms (Procedures)

CREATE [OR REPLACE] PROCEDURE
name [parameters]
IS|AS (Mandatory)
Variables, cursors, etc.; (Optional)
BEGIN (Mandatory)
SQL and PL/SQL statements;
EXCEPTION (Optional)
WHEN exception-handling actions;
END [name]; (Mandatory)

Blocuri anonime si subprograme

Anonymous Blocks (blocuri anonime)	Subprograms (Suprograme)
Blocuri PL/SQL fara nume	Blocuri PL/SQL cu nume
Se compileaza la fiecare executie	Se compileaza o data, numai la creare
Nu se stocheaza in baza de date	Se stocheaza in baza de date
Nu pot fi utilizate(folosite) de catre alte aplicatii	Au nume si de aceea pot fi utilizate de catre alte aplicatii
Nu returneaza valori	Subprogramele numite functii trebuie sa returneze valori
Nu pot avea parametri	Pot avea parametri

Avantajele folosirii subprogramelor

Procedurile si functiile au multe avantaje ca urmare a modularizarii codului:

1. Intretinere usoara:

- ✓ Modificarile este suficient sa se faca o data pentru a imbunatati mai multe aplicatii si pentru a minimiza testele

2. Reutilizarea codului:

- ✓ Subprogramele sunt puse intr-un singur loc.
- ✓ Odata compilate si validate, pot utilizate si reutilizate in oricate aplicatii.

Avantajele folosirii subprogramelor

3. Imbunatateste securitatea datelor:

- ✓ Accesul indirect la obiectele bazei de date este permis de acordarea privilegiilor de securitate asupra subprogramelor.
- ✓ Implicit, subprogramele functioneaza cu privilegiile proprietarului, nu cu cele ale utilizatorului.

4. Integritatea datelor:

- ✓ Actiunile pot fi grupate intr-un bloc si sunt executate impreuna sau deloc.

Avantajele folosirii subprogramelor

5. *Imbunatatirea performantelor:*

- ✓ Puteti refolosi codul PL/SQL compilat, care este stocat in **zona cache SQL de partajare**.
- ✓ Subsecventele care apeleaza subprogramul evita recompilarea codului.
- ✓ De asemenea, multi utilizatori pot imparti o singura copie a codului unui subprogram din memorie.

6. *Imbunatatirea claritatii codului:*

- ✓ Prin utilizarea unor nume si conventii adecvate pentru a descrie actiunea subprogramului, puteti reduce necesarul de comentarii si spori claritatea codului.

In concluzie, procedurile si functiile:

- Sunt blocuri PL/SQL denumite
- Sunt numite subprograme PL/SQL
- Au structuri de bloc asemanatoare blocurilor anonime:
 - Parametri optionali
 - Sectiune declarativa optionala (dar cuvantul cheie **DECLARE** se schimba in **IS** sau **AS**)
 - Sectiune executabila obligatorie
 - Sectiune optionala de manipulare a exceptiilor

Proceduri

- *O procedura este un bloc PL/SQL cu nume care poate accepta parametri*
- In general, puteti folosi o procedura pentru a realiza o actiune (uneori numita „efect secundar”)
- O procedura este compilata si stocata in baza de date ca un obiect din schema
 - Este prezentata in **USER_OBJECTS** ca un obiect de tip **PROCEDURE**
 - Mai multe detalii sunt prezentate in **USER_PROCEDURES**
 - Codul PL/SQL detaliat este in **USER_SOURCE**

Sintaxa pentru crearea procedurilor

```
CREATE [OR REPLACE] PROCEDURE  
procedure_name  
[(parameter1 [mode1] datatype1,  
  parameter2 [mode2] datatype2, . . .)]  
IS|AS  
  
procedure_body;
```

- Parametrii sunt optionalii
- Modul implicit este **IN**
- Tipul de date poate fi atat **explicit** (de exemplu **VARCHAR2**) cat si **implicit** cu **%TYPE**
- Corpul de instructiuni este asemanator cu cel al unui bloc anonim.
- Folositi **CREATE PROCEDURE** urmat de nume, parametrii optionalii si unul din cuvintele cheie **IS** sau **AS**
- Adaugati optiunea **OR REPLACE** pentru a suprascrie o procedura existenta
- Scrieti un bloc PL/SQL care contine variabile locale, un **BEGIN** si un **END** (sau **END procedure_name**)

Sintaxa pentru crearea procedurilor

```
CREATE [OR REPLACE] PROCEDURE  
  procedure_name  
  [(parameter1 [mode] datatype1,  
    parameter2 [mode] datatype2, ...)]  
IS|AS  
[local_variable_declarations; ...]  
BEGIN  
  
...  
END [procedure_name];
```

Exemplu

In urmatorul exemplu, procedura **add_dept** insereaza un departament nou care are **deptno = 280** si **dname = ST-Curriculum**. Procedura declara in sectiunea declarativa doua variabile **v_dept_id** si **v_dept_name**.

```
CREATE OR REPLACE PROCEDURE  
  add_dept IS  
v_dept_id dept.deptno%TYPE;  
v_dept_name dept.dname%TYPE;
```

BEGIN

add_dept;

v_dept_id :=280;

v_dept_name :='ST-Curriculum';

INSERT INTO dept(deptno,dname)

VALUES(v_dept_id,v_dept_name);

**DBMS_OUTPUT.PUT_LINE('Inserted
'||SQL%ROWCOUNT ||'row');**

END;

Partea declarativa a procedurii incepe imediat dupa declararea procedurii si nu incepe cu cuvantul cheie **DECLARE**.

Aceasta procedura foloseste atributul de cursor **SQL%ROWCOUNT** pentru a verifica daca randul a fost inserat cu succes.

SQL%ROWCOUNT returneaza 1 in acest caz.

ORACLE Application Express

Home Application Builder ▾ SQL Workshop ▾ Team Development ▾ Administration ▾

SQL Workshop SQL Commands

Rows

10

Save

Run

```
create or replace procedure ADD_DEPT is
v_dept_id dept.deptno%TYPE;
v_dept_name dept.dname%TYPE;
BEGIN
    add_dept;
    v_dept_id :=380;
    v_dept_name :='ST-Curriculum';
    INSERT INTO dept(deptno, dname)
    VALUES(v_dept_id,v_dept_name);
    DBMS_OUTPUT.PUT_LINE('Inserted ' || SQL%ROWCOUNT || 'row');
END;
```

Proceduri care se apeleaza

O procedura se poate apela din:

1. *Un bloc anonim*
2. *Alta procedura*
3. *O aplicatie apelanta*

Observatie:

Nu puteti apela o procedura din interiorul unei instructiuni SQL cum ar fi **SELECT**.

Pentru a executa o procedura in **Oracle Application Express**, scrieti si rulati un mic bloc anonim care apeleaza procedura.

Exemplu:

CREATE OR REPLACE PROCEDURE

add_dept IS ...

BEGIN

add_dept;

SELECT deptno, dname

FROM dept

WHERE deptno=280;

END;

Instructiunea **SELECT** din final confirma faptul ca randul a fost inserat cu succes.

Corectarea erorilor in instructiunile CREATE PROCEDURE

- Daca sunt erori de compilare, **Application Express** le afiseaza in partea de afisare a ferestrei de comenzi **PL/SQL**.
- Trebuie sa editam codul sursa pentru a corecta erorile.
- Procedura este inca creata chiar daca contine erori.

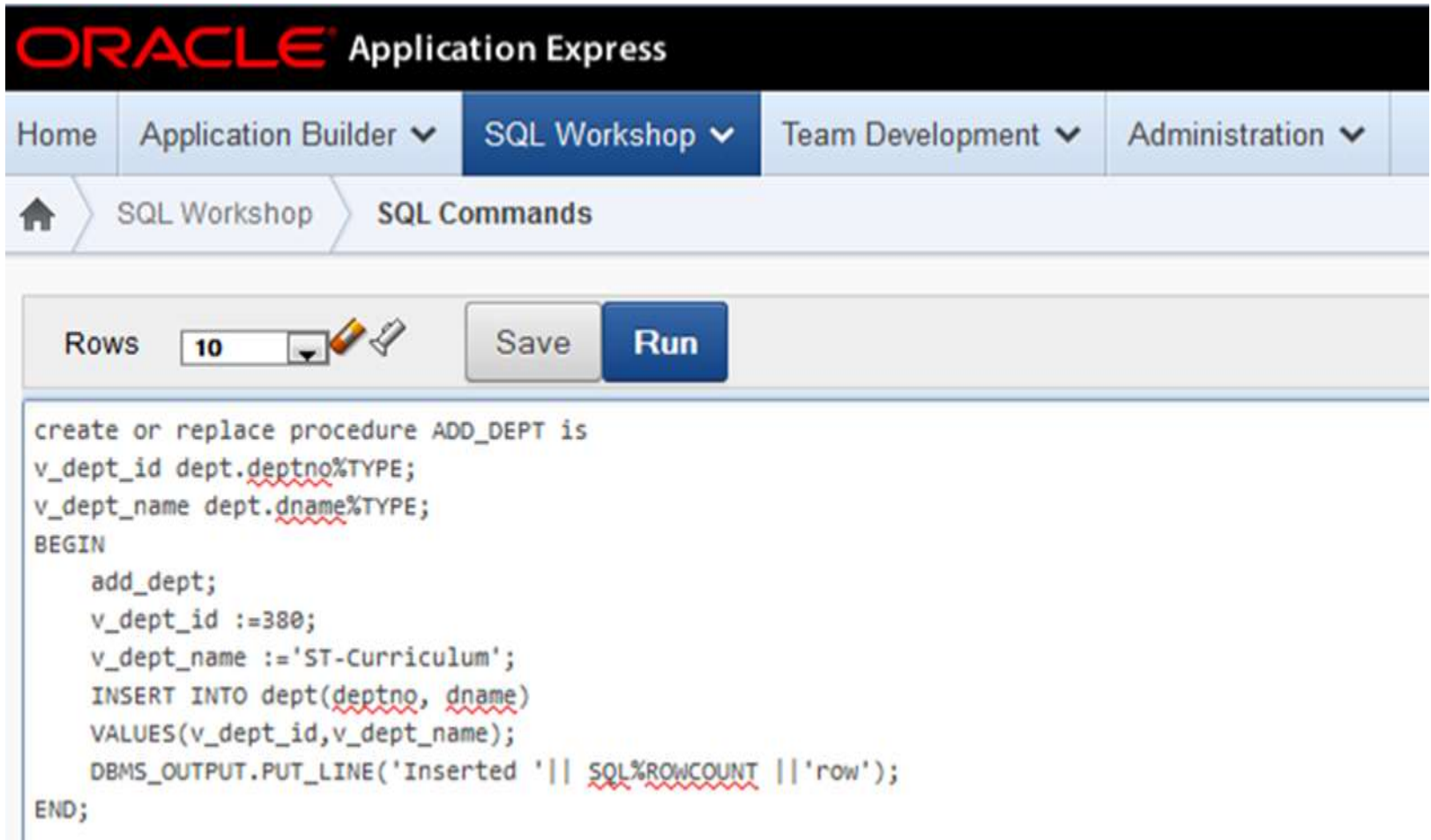
Corectarea erorilor in instructiunile CREATE PROCEDURE

Dupa ce am corectat erorile, este necesar sa recream procedura.

Sunt doua modalitati de a face acest lucru:

1. Folosirea unei instructiuni **CREATE or REPLACE PROCEDURE** pentru a rescrie codul existent (metoda cea mai folosita)
2. Eliminarea procedurii mai intai (**DROP**) si apoi executarea instructiunii **CREATE PROCEDURE** (mai putin folosita)

- După ce procedura a fost creată cu succes, definiția ei ar trebui salvată dacă doriți să-i modificați codul ulterior.



The screenshot displays the Oracle Application Express interface. At the top, the 'ORACLE Application Express' logo is visible. Below it, a navigation menu includes 'Home', 'Application Builder', 'SQL Workshop', 'Team Development', and 'Administration'. The 'SQL Workshop' section is active, showing 'SQL Commands'. A toolbar contains a 'Rows' dropdown set to '10', a 'Save' button, and a 'Run' button. The main area contains the following PL/SQL code:

```
create or replace procedure ADD_DEPT is
v_dept_id dept.deptno%TYPE;
v_dept_name dept.dname%TYPE;
BEGIN
    add_dept;
    v_dept_id :=380;
    v_dept_name :='ST-Curriculum';
    INSERT INTO dept(deptno, dname)
    VALUES(v_dept_id,v_dept_name);
    DBMS_OUTPUT.PUT_LINE('Inserted ' || SQL%ROWCOUNT || 'row');
END;
```

In **Application Express** in fereastra de comenzi **SQL** faceti click pe **SAVE** si introduceti un nume si o descriere optionala pentru codul vostru.

The screenshot displays the Oracle Application Express interface. At the top, the title bar reads "ORACLE Application Express". Below it is a navigation menu with tabs for "Home", "Application Builder", "SQL Workshop", "Team Development", and "Administration". The "SQL Workshop" tab is active, and the sub-tab "SQL Commands" is selected. The main area contains a toolbar with a "Rows" dropdown set to "10", a "Save" button, and a "Run" button. Below the toolbar is a text area containing the following SQL code:

```
create or replace procedure ADD_DEPT is
v_dept_id dept.deptno%TYPE;
v_dept_name dept.dname%TYPE;
BEGIN
  add_dept;
  v_dept_id :=280;
  v_dept_name :='ST-Curriculum';
  INSERT INTO dept(deptno, dname)
  VALUES(v_dept_id,v_dept_name);
  DBMS_OUTPUT.PUT_LINE('Inserted '|| SQL%ROWCOUNT ||'row');
END;
```

At the bottom of the interface, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". A "Procedure created." message is visible at the bottom left. On the right side, a "Save SQL" dialog box is open, showing the "Name" field with the value "add dept" and the "Description" field with the text "Procedura care adauga un rand in departamentul 280". The dialog box has "Cancel" and "Save" buttons at the bottom.

Puteti vizualiza si reincarca codul ulterior facand click pe butonul **Saved SQL** in fereastra de comenzi SQL.

ORACLE Application Express

Home Application Builder SQL Workshop Team Development Administration

SQL Workshop SQL Commands

Schema ORACLE_S

Rows 10

Save

Run

```
CREATE OR REPLACE PROCEDURE add_dept IS
v_dept_id dept.deptno%TYPE;
v_dept_name dept.dname%TYPE;

BEGIN
  add_dept;
  v_dept_id :=280;
  v_dept_name :='ST-Curriculum';
  INSERT INTO dept(deptno,dname)
  VALUES(v_dept_id,v_dept_name);
  DBMS_OUTPUT.PUT_LINE('Inserted '||SQL%ROWCOUNT ||'row');
END;
```

Results Explain Describe Saved SQL History

Owner - All Users -

Find

Rows 10

Go

Delete Checked

<input type="checkbox"/>	Owner	Name	Description	SQL
<input type="checkbox"/>	ADRIAN.RUNCEANU@GMAIL.COM	add_dept	Procedura care adauga un rand in departamentul 280	CREATE OR REPLACE PROCEDURE add_dept IS v_dept_id dept.deptno%TYPE; v_dept_name dept.dname%TYPE; BEGIN add_dept; v_

Cuprins

- 1. Determinarea domeniului variabilelor - recapitulare**
- 2. Proceduri și funcții**
- 3. Folosirea parametrilor în proceduri**

Folosirea parametrilor in proceduri

- Pentru a face procedurile mai flexibile, este important sa oferim date variate procedurii prin intermediul parametrilor de intrare.
- Rezultatele calculate pot fi returnate prin folosirea parametrilor **OUT** sau **IN OUT**.

Folosirea parametrilor in proceduri

Ce sunt parametrii?

- *Parametrii transmit si comunica date intre apelant si subprogram.*
- Parametri sunt *variabile speciale ale caror valori de intrare sunt initializate de mediul apelant atunci cand subprogramul este apelat, iar rezultatele sunt returnate mediului apelant.*

Folosirea parametrilor in proceduri

Ce sunt parametrii?

- Prin conventie, parametrii sunt numiti adesea cu prefixul „**p_**”.
- Parametri sunt referiti de obicei ca argumente.
- In orice caz, argumentele sunt mult mai adecvate ca valori reale atribuite variabilelor parametri atunci cand subprogramul este apelat.
- *Chiar daca parametrii sunt un fel de variabile, parametrii de tip **IN** se comporta ca si constante si nu pot fi schimbati de subprogram.*

Crearea procedurilor cu parametri

Urmatorul exemplu prezinta o procedura cu 2 parametri.

Se creeaza procedura **raise_salary** in baza de date. Apoi se executa procedura transmitandu-i valorile 176 si 10 pentru cei doi parametri.

```
CREATE OR REPLACE PROCEDURE raise_salary  
(p_id IN my_emp.empno%TYPE, p_percent IN  
NUMBER)
```

```
IS
```

```
BEGIN
```

```
UPDATE my_employees
```

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

```
WHERE empno = p_id;
```

```
END raise_salary;
```

```
BEGIN raise_salary(176,10); END;
```

Apelarea procedurilor cu parametri

- Se foloseste numele procedurii si valorile parametrilor ca in exemplul anterior.
- Argumentele trebuie sa fie in aceeasi ordine in care sunt declarate in procedura.
- Pentru ca o procedura sa fie apelata de catre alta procedura, se foloseste un apel direct in partea executabila a blocului.

Exemplu

CREATE OR REPLACE PROCEDURE

- **process_employees**

IS

CURSOR emp_cursor IS

SELECT empno

FROM my_employees;

BEGIN

FOR v_emp_rec IN emp_cursor

LOOP

raise_salary(v_emp_rec.empno, 10);

END LOOP;

COMMIT;

END process_employees;

Tipuri de parametri

- Sunt doua tipuri de parametri:
 1. **formali**
 2. **actuali**
- Un parametru cu nume declarat in antetul procedurii este numit *parametru formal*.
- Parametrul corespunzator de la apel se numeste *parametru actual*.

Tipuri de parametri

In urmatorul exemplu puteti spune ce parametru este formal si ce parametru este actual?

```
CREATE OR REPLACE PROCEDURE
```

```
fetch_emp
```

```
(p_emp_id IN employees.empno%TYPE)
```

```
IS ...
```

```
END;
```

```
/* Acum se face apelul procedurii dintr-un bloc  
anonim */
```

```
BEGIN fetch_emp(v_emp_id); END;
```

Parametrii formali

Parametrii formali sunt variabile care sunt declarate in lista de parametri a specificatiei subprogramului.

Parametrii formali

In urmatorul exemplu, in procedura ***raise_sal***, identificatorii ***p_id*** si ***p_sal*** reprezinta parametri formali.

```
CREATE PROCEDURE raise_sal (p_id IN  
NUMBER, p_sal IN NUMBER) IS  
BEGIN...  
END raise_sal;
```

Observati ca tipurile de date ale parametrilor formali nu au dimensiuni.

De exemplu, ***p_sal*** este de tip NUMBER si nu NUMBER(6,2).

Parametrii actuali

Parametrii actuali pot fi:

- *literali,*
- *variabile*
- *sau expresii care apar in lista de parametri a unui subprogram apelat.*

Parametrii actuali

In urmatorul exemplu se apeleaza ***raise_sal*** unde variabila ***a_emp_id*** este parametrul actual pentru parametrul formal ***p_id***.

```
a_emp_id := 100;  
raise_sal(a_emp_id, 2000);
```

Parametrii actuali:

Sunt asociati cu parametrii formali cand se apeleaza subprogramul

Pot fi expresii – de exemplu:

```
raise_sal(a_emp_id, v_raise+100);
```

Tipuri de parametri

- *Parametrii formali si parametrii actuali trebuie sa fie de tipuri de date compatibile.*
- Daca este necesar, inainte de atribuirea de valori, **PL/SQL** converteste tipul de date al valorii parametrului actual la cel al parametrului formal.
- Puteti gasi tipurile de date necesare folosind comanda **DESCRIBE** `proc_name`.



Întrebări?