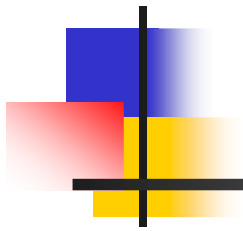


CURS 11



PL/SQL IV Modularizarea programelor



Modularitate – subprograme și pachete

- Modularitatea permite spargerea unei aplicații în module bine definite.
- Prin rafinări succesive se poate reduce o problemă complexă la o mulțime de probleme simple care au soluții simple de implementat
- PL/SQL permite acest lucru prin unitățile de program, care includ *blocuri*, *subprograme* și *pachete*



Subprograme

PL/SQL admite două tipuri de subprograme - *proceduri* și *functii*

- Sunt blocuri PL/SQL denumite
- Au structura similară blocurilor anonime:
 - Secțiune declarativa - opțională
 - Secțiune executabilă – obligatorie
 - Secțiune de manipulare a excepțiilor – opțională
- Avantajele utilizării subprogramelor:
 - Extensibilitate – permit construcții în SQL care să răspundă diferitelor nevoi
 - Modularitate
 - Pot fi reutilizate
 - Simplifica mentenanța
 - Permit abstractizarea



Proceduri - Sintaxa generală

```
[CREATE [OR REPLACE]]
PROCEDURE nume_proc [param[, param]...]
    [AUTHID {DEFINER | CURRENT USER}]
    {IS|AS}
    [PRAGMA
    AUTONOMOUS_TRANSACTION;]
    [declaratii locale]
BEGIN
    Instructiuni in sectiunea executabila
[EXCEPTION
    handler pt exceptii]
END [nume_proc]
```

Unde param este de următoarea formă:

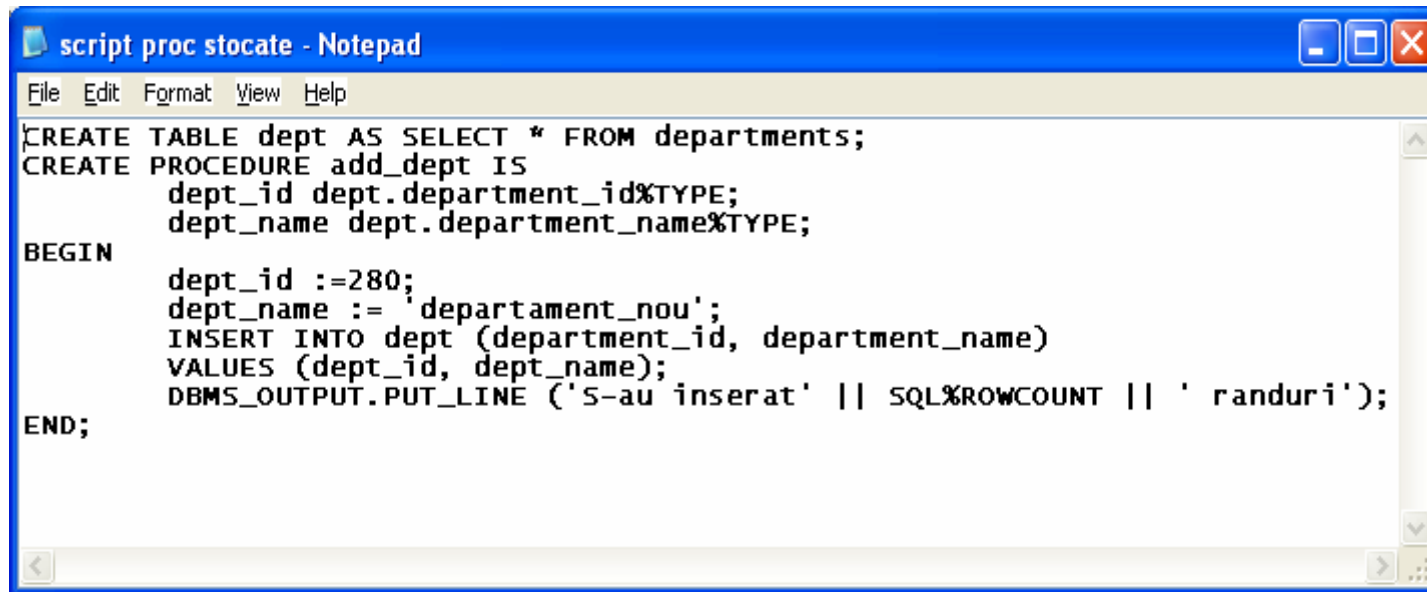
Nume_para [IN | OUT | IN OUT] tip_data
[{: = | DEFAULT} expresie]

NOTA: pentru tipul de dată nu se specifică dimensiunea

Unde:

- CREATE – permite crearea de proceduri independente stocate in baza de date (proceduri stocate)
 - Pentru a fi executata instructiunea CREATE PROCEDURE in cadrul unui program se foloseste SQL dinamic
- AUTHID – determină daca o procedura stocată se execută cu privilegiile deținute de proprietarul său (implicit este userul curent) si dacă referințele necalificate la obiectele schema sunt rezolvate in schema proprietarului
- PRAGMA AUTONOMOUS_TRANSACTION – specifica compilatorului PL/SQL să marcheze procedura ca autonomă
 - Se permite suspendarea tranzactiei in curs, se executa operatii SQL care se valideaza sau se abandoneaza dupa care se reia tranzactia suspendata

Proceduri stocate - Exemplu



```
script proc stocate - Notepad
File Edit Format View Help
CREATE TABLE dept AS SELECT * FROM departments;
CREATE PROCEDURE add_dept IS
    dept_id dept.department_id%TYPE;
    dept_name dept.department_name%TYPE;
BEGIN
    dept_id := 280;
    dept_name := 'departament_nou';
    INSERT INTO dept (department_id, department_name)
    VALUES (dept_id, dept_name);
    DBMS_OUTPUT.PUT_LINE ('S-au inserat' || SQL%ROWCOUNT || ' randuri');
END;
```

- Procedurile stocate sunt obiecte in baza de date
- La crearea oricarui obiect se insereaza date in tabelul user_object
 SELECT object_name, object_type FROM user_objects
 WHERE object_type in ('TABLE', 'PROCEDURE');

OBJECT_NAME	OBJECT_TYPE
ADD_DEPT	PROCEDURE
ADD_JOB_HISTORY	PROCEDURE
AFISAREE	PROCEDURE
AGENDATEL	TABLE
AGENDA_TELFONICA	TABLE
AGENTEL	TABLE
AGENTELE	TABLE
ANG	PROCEDURE
ANGAJAT	PROCEDURE
ANGAJATI33	TABLE
AUDIT_NRST_CP	TABLE
A_AGAJATI	TABLE
CARTI_OL	TABLE
COPIE_EMP	TABLE
OBJECT_NAME	OBJECT_TYPE
COUNT_EMPLOYEE	PROCEDURE
CUC1	PROCEDURE
CURSURI_UNG	TABLE
C_C	TABLE
C_R	TABLE
C_T	TABLE
DEPARTMENTS	TABLE
DEPT	TABLE

- Sursa codului corespunzator procedurii este memorata in tabelul user_source

Introducere instrucțiuni:

```
SELECT * FROM user_source WHERE name='ADD_DEPT';
```

Execuție

Salvare script

Ștergere ecran

Renunțare

NAME	TYPE	LINE	TEXT
ADD_DEPT	PROCEDURE	1	PROCEDURE add_dept IS
ADD_DEPT	PROCEDURE	2	dept_id dept.department_id%TYPE;
ADD_DEPT	PROCEDURE	3	dept_name dept.department_name%TYPE;
ADD_DEPT	PROCEDURE	4	BEGIN
ADD_DEPT	PROCEDURE	5	dept_id :=280;
ADD_DEPT	PROCEDURE	6	dept_name := 'departament_nou';
ADD_DEPT	PROCEDURE	7	INSERT INTO dept (department_id, department_name)
ADD_DEPT	PROCEDURE	8	VALUES (dept_id, dept_name);
ADD_DEPT	PROCEDURE	9	DBMS_OUTPUT.PUT_LINE ('S-au inserat' SQL%ROWCOUNT ' randuri');
ADD_DEPT	PROCEDURE	10	END;

10 înregistrări selectate.

Invocarea unei proceduri stocate

Introducere instrucțiuni:

```
BEGIN
  add_dept;
END;
/
SELECT department_id, department_name FROM dept WHERE
department_id=280;
```

Execuție

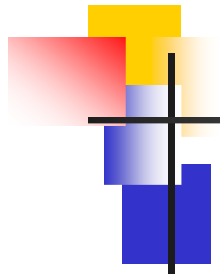
Salvare script

Ștergere ecran

Renunțare

Procedură PL/SQL încheiată cu succes.

DEPARTMENT_ID	DEPARTMENT_NAME
280	Educatie
280	departament_nou



Proceduri locale

- Procedura locală nu este memorată în baza de date dar este utilizată în cadrul unui program
- Dacă se omite utilizarea **CREATE OR REPLACE**, atunci procedura devine o procedură locală.
- Subprogramele locale sunt definite în secțiunea declarativă a unui bloc PL/SQL
- **Apelul unei proceduri** se face din interiorul unui bloc PL/SQL ca o declarație executabilă

`nume_procedura(argumente);`



Proceduri locale - Exemplu

DECLARE

PROCEDURE award_bonus (emp_id NUMBER) IS

 bonus REAL;

 comm_missing EXCEPTION;

BEGIN

 SELECT comm * 0.15 INTO bonus FROM emp WHERE empno = emp_id;

 IF bonus IS NULL THEN

 RAISE comm_missing;

 ELSE

 UPDATE payroll SET pay = pay + bonus

 WHERE empno = emp_id;

 END IF;

EXCEPTION

 WHEN comm_missing THEN

 ...

END award_bonus;

BEGIN

award_bonus(105);

END



Subprograme - Funcții

- funcția – subprogram care calculează și întoarce o valoare
- Are o structură similară procedurilor
 - suplimentar conține o clauză RETURN

[CREATE [OR REPLACE]]

FUNCTION nume_func [arg1 [mod] tip_data[, arg2 ...]...]

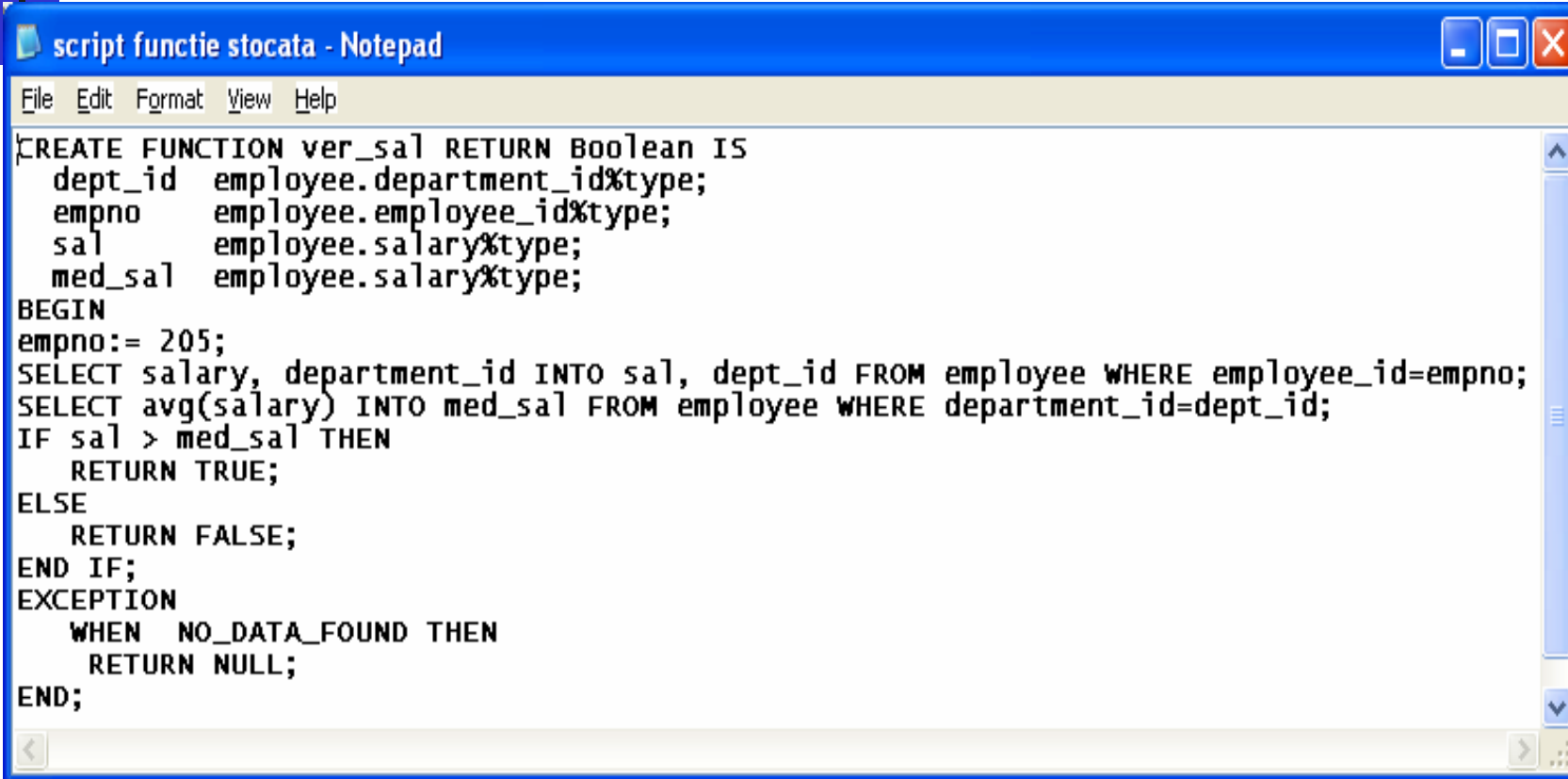
RETURN tip_data

IS |AS

corpul funcției

- Modul unui argument specifică tipul parametrului. Se vor declara numai parametrii de tip IN

Exemplu



```
script functie stocata - Notepad
File Edit Format View Help
CREATE FUNCTION ver_sal RETURN Boolean IS
  dept_id employee.department_id%type;
  empno    employee.employee_id%type;
  sal      employee.salary%type;
  med_sal  employee.salary%type;
BEGIN
  empno:= 205;
  SELECT salary, department_id INTO sal, dept_id FROM employee WHERE employee_id=empno;
  SELECT avg(salary) INTO med_sal FROM employee WHERE department_id=dept_id;
  IF sal > med_sal THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
END;
```

Exemplu – Invocarea functiei

Ecran de lucru

Fișier sau URL:

Browse...

Încărcare script

Introducere instrucțiuni:

```
set serveroutput on
BEGIN
  IF (ver_sal is NULL) THEN
    DBMS_OUTPUT.PUT_LINE ('Functia a intalnit o exceptie');
  ELSIF (ver_sal) THEN
    DBMS_OUTPUT.PUT_LINE ('Salariul > media');
  ELSE
    DBMS_OUTPUT.PUT_LINE ('Salariul < media');
  END IF;
END;
```

Execuție

Salvare script

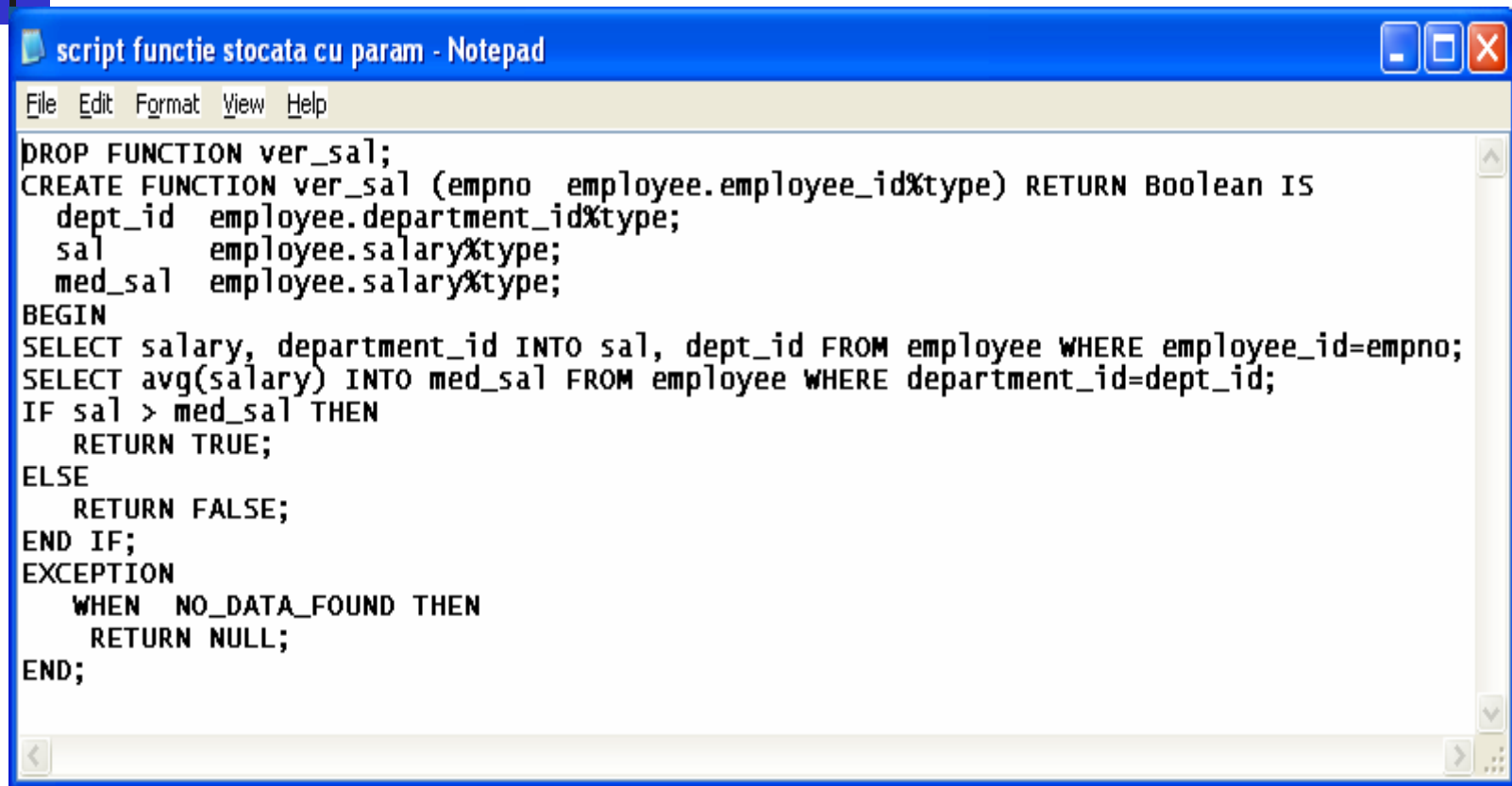
Ștergere ecran

Renunțare

Salariul > media

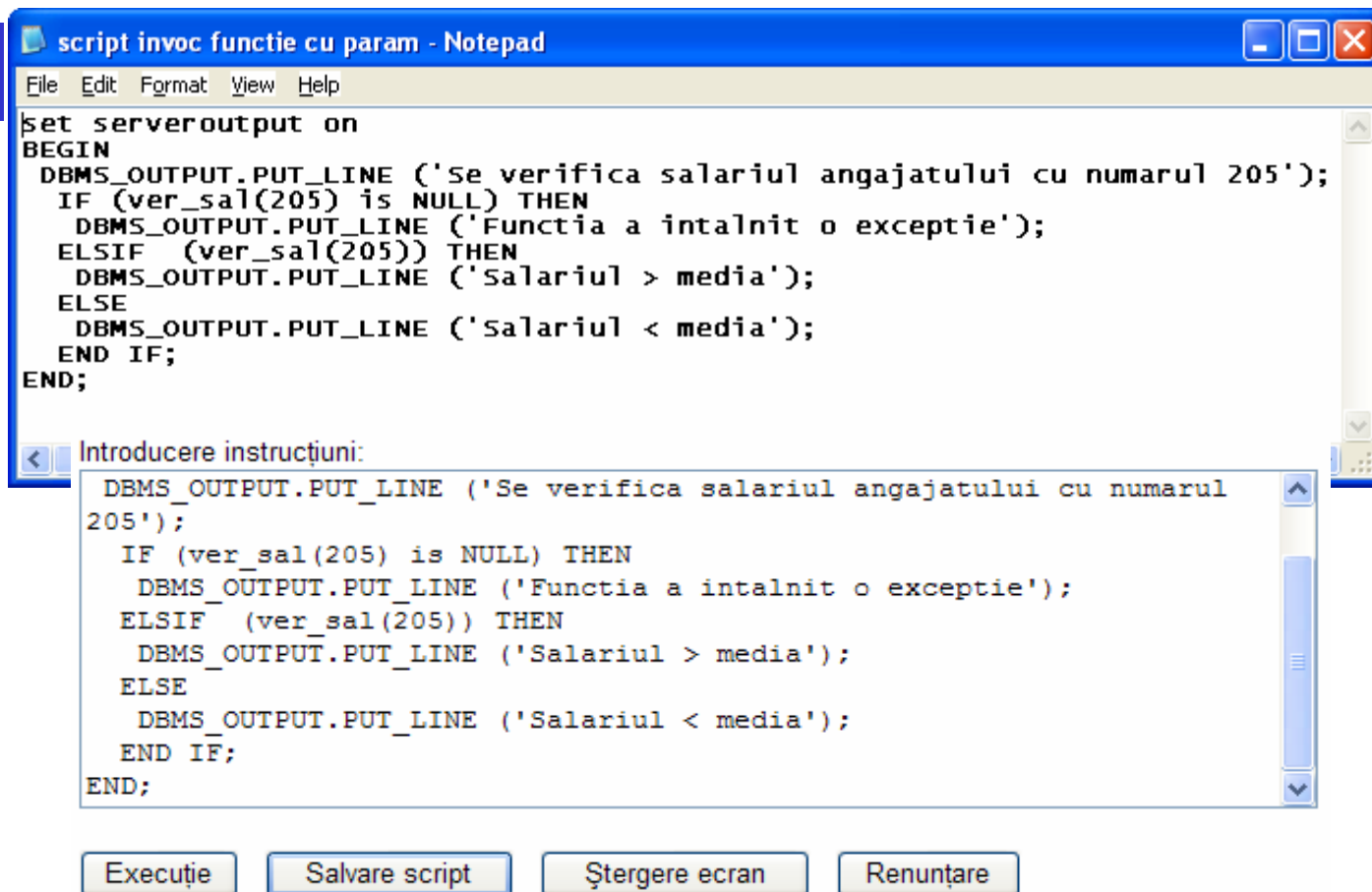
Procedură PL/SQL încheiată cu succes.

Transmiterea parametrilor catre o functie



```
script functie stocata cu param - Notepad
File Edit Format View Help
DROP FUNCTION ver_sal;
CREATE FUNCTION ver_sal (empno employee.employee_id%type) RETURN Boolean IS
    dept_id employee.department_id%type;
    sal      employee.salary%type;
    med_sal  employee.salary%type;
BEGIN
    SELECT salary, department_id INTO sal, dept_id FROM employee WHERE employee_id=empno;
    SELECT avg(salary) INTO med_sal FROM employee WHERE department_id=dept_id;
    IF sal > med_sal THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
```

... si invocarea unei functii cu parametri



```
script invoc functie cu param - Notepad
File Edit Format View Help
set serveroutput on
BEGIN
  DBMS_OUTPUT.PUT_LINE ('Se verifica salariul angajatului cu numarul 205');
  IF (ver_sal(205) is NULL) THEN
    DBMS_OUTPUT.PUT_LINE ('Functia a intalnit o exceptie');
  ELSIF (ver_sal(205)) THEN
    DBMS_OUTPUT.PUT_LINE ('Salariul > media');
  ELSE
    DBMS_OUTPUT.PUT_LINE ('Salariul < media');
  END IF;
END;
```

Introducere instrucțiuni:

```
DBMS_OUTPUT.PUT_LINE ('Se verifica salariul angajatului cu numarul
205');
  IF (ver_sal(205) is NULL) THEN
    DBMS_OUTPUT.PUT_LINE ('Functia a intalnit o exceptie');
  ELSIF (ver_sal(205)) THEN
    DBMS_OUTPUT.PUT_LINE ('Salariul > media');
  ELSE
    DBMS_OUTPUT.PUT_LINE ('Salariul < media');
  END IF;
END;
```

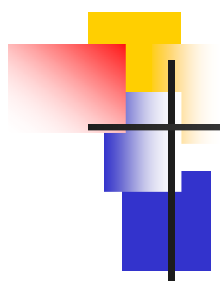
Execuție Salvare script Ștergere ecran Renunțare

Se verifica salariul angajatului cu numarul 205
Salariul > media
Procedură PL/SQL încheiată cu succes.



Controlul efectelor produse de subprogramele PL/SQL

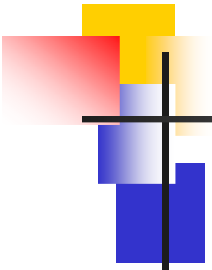
- Pentru a fi apelată dintr-o instrucțiune SQL o funcție stocată trebuie să respecte următoarele reguli
 - Când este apelată dintr-un SELECT sau una din instrucțiunile INSERT, UPDATE sau DELETE paralelizate funcția nu poate modifica nici un tabel al bazei de date
 - Când este apelată dintr-o instrucțiune INSERT, UPDATE sau DELETE funcția nu poate interoga sau modifica nici un tabel al bazei de date, modificat prin instrucțiunea apelantă
 - Dacă este apelată în orice instrucțiune DML din SQL funcția nu poate conține:
 - instrucțiuni de control al tranzacțiilor (COMMIT sau ROLLBACK)
 - Instrucțiuni de control al sesiunii de lucru (SET ROLE)
 - Instrucțiuni de control al sistemului (ALTER SYSTEM)
 - În acest caz nu poate executa instrucțiuni DDL, deoarece de multe ori acestea sunt urmate de un COMMIT automat



Pachete

- construcții PL/SQL care permit obiectelor înrudite să fie stocate împreună
- conține două părți distincte:
 - **specificația** - conține informații despre conținutul pachetului și nu secvențe de cod.
 - **corpul pachetului.**
 - Fiecare din acestea **sunt memorate separat.**
- Un pachet poate fi **doar stocat** (NU poate fi o construcție locală într-un bloc),
 - NU se apelează niciodată pachetul (procedurile/functiile definite într-un pachet sunt apelate).

Specificatia unui pachet (package header)



```
CREATE [OR REPLACE] PACKAGE nume_pachet {IS| AS}  
    specificatie de procedura |specificatie de functie |  
    declaratii_variabibile |definitii de tipuri | declarare de exceptii |  
    declaratii de cursoare
```

```
END [nume_pachet];
```

```
CREATE OR REPLACE PACKAGE ex_pkg IS  
    v_ex NUMBER :=0.1;  
    PROCEDURE reset_comm(new_comm NUMBER);  
END ex_pkg;
```



Corpul pachetului (package Body)

- obiect separat de specificatie in dictionarul de date
- Contine codul corespunzator declarațiilor din header -- nu poate fi compilat in absenta specificatiei.
- Specificatia pentru proceduri sau functii trebuie sa fie aceeași atât în header cât si in body.
- Corpul pachetului poate contine elemente care nu apar in specificatie. Acestea se numesc **elemente private** ale pachetului. Un element privat nu poate fi referit in afara pachetului deoarece el nu apare in specificatie.

- Sintaxa generala:

```
CREATE OR REPLACE PACKAGE BODY nume_pachet {IS|AS}  
    ...  
    [BEGIN]  
    ...  
END nume_pachet];
```



```
CREATE OR REPLACE PACKAGE BODY ex_pkg IS
```

```
FUNCTION validate (comm NUMBER) RETURN BOOLEAN IS
    max_comm employee.commission_pct%TYPE;
BEGIN
    SELECT max(commission_pct) into max_comm FROM employee;
    RETURN (comm BETWEEN 0.0 and max_comm);
END validate;
```

```
PROCEDURE reset_comm(new_comm NUMBER) IS
BEGIN
    IF validate(new_comm) THEN
        v_ex:= new_comm;
        ELSE RAISE_APPLICATION_ERROR(-20210, 'Comision eronat');
    END IF;
END reset_comm;
END ex_pkg;
```

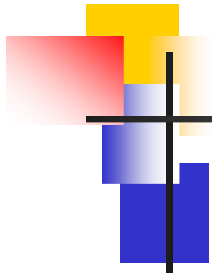


Domeniul de vizibilitate al pachetelor

- Orice obiect declarat in header-ul pachetului este vizibil in afara acestuia.
 - acest lucru poate fi util in declararea variabilelor globale.
- Obiectele pot fi adresate prin prefixarea numelui cu numele pachetului.

Exemplu: `DBMS_OUTPUT.PUT_LINE('hello');`

- Apelul procedurii este similar apelului unei proceduri de sine statatoare.



■ Suprapunerea subprogramelor

- In cadrul unui pachet, procedurile si functiile pot fi suprapuse.
 - Aceasta inseamna ca pot fi mai multe proceduri sau functii cu acelasi nume, dar -obligatoriu- cu parametri diferiti.

■ Pachete si dependente

- Corpul unui pachet depinde de **header** și de **tabelele referite**.
- Header-ul pachetului nu depinde de nimic (aceasta reprezintă un avantaj). Ca urmare se poate modifica corpul unui pachet fără a schimba header-ul acestuia (ascunderea informatiei!).
- Dacă se schimbă antetul unui pachet automat este invalidat corpul acestuia.



Eliminarea pachetelor

- DROP PACKAGE nume_pachet
- DROP PACKAGE BODY nume_pachet



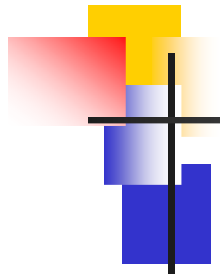
Vizualizarea pachetelor în dicționarul de date

- Codul sursa pentru pachetele PL/SQL este pastrat si poate fi vizualizat in tabelele USER_SOURCE respectiv ALL_SOURCE din dicționarul de date
- Pentru a vedea specificația:

```
SELECT text  
FROM user_source  
WHERE name= 'EX_PKG' and type='PACKAGE';
```

- Pentru a vizualiza corpul pachetului se folosește:

```
SELECT text  
FROM user_source  
WHERE name= 'EX_PKG' and type='PACKAGE BODY';
```

Reguli pentru scrierea pachetelor

- Pachetele se construiesc pentru scopuri cu un grad sporit de generalitate
- Specificația pachetului se definește înaintea corpului acestuia
- Specificația trebuie sa conțină numai acele construcții care se doresc a fi publice
- Schimbarea specificației pachetului necesita recompilarea tuturor subprogramelor referite
- Specificația unui pachet trebuie sa conțină minimul posibil de construcții



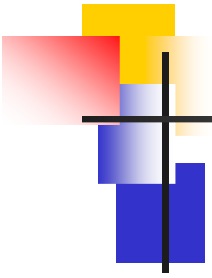
Avantajele utilizării pachetelor

- Modularitate – încapsulează construcțiile înrudite
- Intreținere ușoară – păstrează laolaltă funcționalitățile între care există legături logice
- Proiectarea ușoară a aplicației – codifica și compilează separat specificația și corpul pachetului
- Ascunde o parte a informației:
 - Numai specificația pachetului este vizibilă și accesibilă aplicațiilor
 - Construcțiile private din interiorul corpului pachetului sunt invizibile și inaccesibile
 - Tot codul este ascuns în corpul pachetului
- Adaugă funcționalități – persistența variabilelor și cursorilor
- Performanțe mai bune
 - Intregul pachet este încărcat în memorie la prima referire
 - Există o singură copie în memorie pentru toți utilizatorii
 - Ierarhia de dependente este simplificată



Pachete preconstruite în Oracle

- Sunt furnizate împreună cu serverul Oracle
- Extind funcționalitățile bazei de date
- Permit accesul la anumite caracteristici SQL care în mod normal sunt restricționate pentru PL/SQL
- Pot fi folosite pentru crearea unor aplicații proprii sau pentru crearea propriilor proceduri stocate

- 
- DBMS_ALERT – oferă notificări asincrone asupra evenimentelor bazei de date
 - Commit determină trimiterea de mesaje sau alerte
 - DBMS_LOCK – utilizat pentru a solicita, converti sau elibera blocări prin intermediul serviciilor Oracle Lock Manager
 - DBMS_SESSION – permite programelor folosirea instrucțiunii ALTER SESSION SQL , precum și a altor instrucțiuni la nivel de sesiune
 - DBMS_OUTPUT – permite depanarea programelor și bufferingul datelor text
 - HTP – scrie date etichetate HTML în bufferele bazei de date
 - UTL_FILE – permite citirea și scrierea fișierelor text din sistemul de operare
 - UTL_MAIL – permite compunerea și trimiterea mesajelor e-mail
 - DBMS_SCHEDULER – permite programarea și execuția automată a blocurilor PL/SQL, a procedurilor stocate a procedurilor externe sau a programelor executabile