

## SGBD-Anul 3

### Laborator 7 PL/SQL

---

#### Tratarea erorilor

➤ Tratarea erorilor se realizează în secțiunea EXCEPTION a blocului PL/SQL:

```
EXCEPTION
  WHEN nume_excepție1 [OR nume_excepție2 ...] THEN
    secvența_de_instrucțiuni_1;
  [WHEN nume_excepție3 [OR nume_excepție4 ...] THEN
    secvența_de_instrucțiuni_2;]
  ...
  [WHEN OTHERS THEN
    secvența_de_instrucțiuni_n;]
END;
```

➤ Cu ajutorul funcțiilor *SQLCODE* și *SQLERRM* se pot obține codul și mesajul asociate excepției declanșate.

- Codul erorii este:
  - un număr negativ, în cazul unei erori sistem;
  - numărul +100, în cazul excepției *NO\_DATA\_FOUND*;
  - numărul 0, în cazul unei execuții normale (fără excepții);
  - numărul 1, în cazul unei excepții definite de utilizator.

➤ Excepțiile pot fi :

- Interne - se produc atunci când un bloc *PL/SQL* nu respectă o regulă *Oracle* sau depășește o limită a sistemului de exploatare.
  - Predefinite - nu trebuie declarate în secțiunea declarativă și sunt tratate implicit de către *server-ul Oracle*. Ele sunt referite prin nume (*CURSOR\_ALREADY\_OPEN*, *DUP\_VAL\_ON\_INDEX*, *NO\_DATA\_FOUND*, *TOO\_MANY\_ROWS*, *ZERO\_DIVIDE*)
  - Excepțiile interne nepredefinite sunt declarate în secțiunea declarativă și sunt tratate implicit de către *server-ul Oracle*. Ele pot fi gestionate prin clauza *OTHERS*, în secțiunea *EXCEPTION* sau prin [vezi](#) mai jos].
- Externe - definite în partea declarativă a blocului, deci posibilitatea de referire la ele este asigurată. În mod implicit, toate excepțiile externe au asociat același cod (+1) și același mesaj (*USER\_DEFINED\_EXCEPTION*)

Declararea și prelucrarea excepțiilor externe respectă următoarea sintaxă:

```
DECLARE
  nume_excepție EXCEPTION; -- declarare excepție
BEGIN
  ...
  RAISE nume_excepție; --declanșare excepție
  -- codul care urmează nu mai este executat
  ...
EXCEPTION
  WHEN nume_excepție THEN
  -- definire mod de tratare a erorii
  ...
END;
```

➤ Altă metodă pentru tratarea unei erori interne nepredefinite (diferită de folosirea clauzei *OTHERS* drept detector universal de excepții) este utilizarea directivei de compilare (pseudo-instrucțiune)

**PRAGMA EXCEPTION\_INIT.** Această directivă permite asocierea numelui unei excepții cu un cod de eroare intern.

În acest caz, tratarea erorii se face în următoarea manieră:

1) se declară numele excepției în partea declarativă sub forma:

```
nume_excepție EXCEPTION;
```

2) se asociază numele excepției cu un cod eroare standard *Oracle*, utilizând comanda:

```
PRAGMA EXCEPTION_INIT (nume_excepție, cod_eroare);
```

3) se referă excepția în secțiunea de gestiune a erorilor (excepția este tratată automat, fără a fi necesară comanda *RAISE*).

➤ Activarea unei excepții externe poate fi făcută și cu ajutorul procedurii *RAISE\_APPLICATION\_ERROR*, furnizată de pachetul *DBMS\_STANDARD*.

*RAISE\_APPLICATION\_ERROR* poate fi folosită pentru a returna un mesaj de eroare unității care o apelează, mesaj mai descriptiv decât identificatorul erorii. Unitatea apelantă poate fi *SQL\*Plus*, un subprogram *PL/SQL* sau o aplicație *client*.

Procedura are următorul antet:

```
RAISE_APPLICATION_ERROR (numar_eroare IN NUMBER,  
mesaj_eroare IN VARCHAR2, [ {TRUE | FALSE } ] );
```

Atributul *numar\_eroare* este un număr cuprins între -20000 și -20999, specificat de utilizator pentru excepția respectivă, iar *mesaj\_eroare* este un text asociat erorii, care poate avea maximum 2048 octeți.

➤ Informații despre erorile apărute la compilare se pot obține consultând vizualizarea *USER\_ERRORS*.

```
SELECT LINE, POSITION, TEXT  
FROM USER_ERRORS  
WHERE NAME = UPPER('nume');
```

*LINE* specifică numărul liniei în care apare eroarea, dar acesta nu corespunde liniei efective din fișierul text (se referă la codul sursă deplasat în *USER\_SOURCE*). Dacă nu sunt erori, apare mesajul *NO ROWS SELECTED*.

### Exerciții:

1. Creați tabelul *erori* având două coloane: *cod\_eroare* de tip *NUMBER* și *mesaj\_eroare* de tip *VARCHAR2(100)*. Să se scrie un bloc *PL/SQL* care să determine și să afișeze salariatul angajat cel mai recent într-un departament al cărui cod este introdus de către utilizator. Pentru orice eroare apărută, vor fi inserate codul și mesajul erorii în tabelul *erori*.

**Varianta 1** (captarea erorii interne a sistemului)

```
ACCEPT p_cod PROMPT 'Introduceti un cod de departament '
```

```
DECLARE
```

```
eroare_cod NUMBER;
```

```
eroare_mesaj VARCHAR2(100);
```

```
v_dep departments.department_id%TYPE := &p_cod;
```

```
v_emp emp%ROWTYPE;
```

```
BEGIN
```

```
SELECT * INTO v_emp
```

```
FROM emp
```

```
WHERE department_id = v_dep
```

```
AND hire_date = (SELECT MAX(hire_date ) FROM emp
```

```
WHERE department_id = v_dep);
```

```
DBMS_OUTPUT.PUT_LINE(v_emp.last_name || ' ' ||v_emp.salary);
```

```

EXCEPTION
  WHEN OTHERS THEN
    eroare_cod := SQLCODE;
    eroare_mesaj := SUBSTR(SQLERRM,1,100);
    INSERT INTO erori
      VALUES (eroare_cod, eroare_mesaj);
END;

```

**Varianta 2** (definirea unei excepții de către utilizator)

```

DECLARE
  eroare_cod    NUMBER;
  eroare_mesaj  VARCHAR2(100);
  v_dep         departments.department_id%TYPE := &p_cod;
  v_emp         emp%ROWTYPE;
  v_num         NUMBER;
  exceptie      EXCEPTION;
BEGIN
  SELECT COUNT(*) INTO v_num -- (1)
  FROM emp
  WHERE department_id = v_dep
  AND hire_date = (SELECT MAX(hire_date ) FROM emp
                  WHERE department_id = v_dep);
  IF v_num != 1 THEN --(2)
    RAISE exceptie;
  END IF;
  --daca nu s-a declansat exceptia, sigur cererea de mai jos va returna o singura linie
  SELECT * INTO v_emp (3)
  FROM emp
  WHERE department_id = v_dep
  AND hire_date = (SELECT MAX(hire_date ) FROM emp
                  WHERE department_id = v_dep);

  /**** Daca nu aveam instructiunile (1) si (2), s-ar fi ridicat exceptia cu acest IF? ****/
  /* IF SQL%ROWCOUNT >!=1 THEN -- fara (1) si (2) nu se ajunge la aceasta instructiune dacă
                                     --(3) declanșează NO_DATA_FOUND sau TOO_MANY_ROWS
    RAISE exceptie;                 -- ar merge utilizarea lui SQL%ROWCOUNT
                                     --pentru INSERT, UPDATE, DELETE
                                     -- (nu pentru SELECT .. INTO ...)
  END IF;
  *****/

  DBMS_OUTPUT.PUT_LINE(v_emp.last_name || ' ' ||v_emp.salary);
EXCEPTION
  WHEN exceptie THEN
    eroare_cod := -20200;
    eroare_mesaj := 'prea multe sau prea putine linii';
    INSERT INTO erori
      VALUES (eroare_cod, eroare_mesaj);
END;
/

```

## 2. [Excepții interne predefinite]

Creați tabelul *mesaje* având o singură coloană, numită *rezultate*, de tip *varchar2(50)*. Să se scrie un bloc *PL/SQL* prin care să se afișeze numele departamentelor dintr-o anumită locație care au

angajați.

- Dacă rezultatul interogării returnează mai mult decât o linie, atunci să se trateze excepția și să se insereze în tabelul *mesaje* textul „mai multe departamente“.
- Dacă rezultatul interogării nu returnează nici o linie, atunci să se trateze excepția și să se insereze în tabelul *mesaje* textul „nici un departament“.
- Dacă rezultatul interogării este o singură linie, atunci să se insereze în tabelul *mesaje* numele departamentului și managerul acestuia.
- Să se trateze orice altă eroare, inserând în tabelul *mesaje* textul „alte erori au apărut“.

```

SET VERIFY OFF
ACCEPT p_locatie PROMPT 'Introduceti locatia:'
DECLARE
  v_nume_dep dep.department_name%TYPE;
  v_manager  dep.manager_id%TYPE;
  v_locatie  dep.location_id%TYPE:= '&p_locatie';
BEGIN
  SELECT  department_name, manager_id
  INTO    v_nume_dep, v_manager_id
  FROM    dep
  WHERE   location = v_locatie;
  INSERT INTO mesaje (rezultate)
  VALUES (v_nume_dep||'-'||v_manager);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    INSERT INTO mesaje (rezultate)
    VALUES ('nici un departament');
  WHEN TOO_MANY_ROWS THEN
    INSERT INTO mesaje (rezultate)
    VALUES ('mai multe departamente');
  WHEN OTHERS THEN
    INSERT INTO mesaje (rezultate)
    VALUES ('alte erori au aparut');
END;
/
SET VERIFY ON

```

### 3. [Excepții interne nepredefinite]

Dacă există angajați ai unui anumit departament, să se tipărească un mesaj prin care utilizatorul este anunțat că departamentul respectiv nu poate fi șters din baza de date (încălcarea constrângerii de integritate având codul eroare *Oracle -2292*). Dacă nu există, implementați această constrângere înainte de execuția blocului.

```

ALTER TABLE dep
ADD CONSTRAINT pk_dep PRIMARY KEY(department_id);

```

```

ALTER TABLE emp
ADD CONSTRAINT fk_emp_dep FOREIGN KEY (department_id) REFERENCES dep;

```

Ce se întâmplă dacă se încearcă ștergerea unui departament în care lucrează angajați?

```

DELETE FROM dep
WHERE department_id=30;
--apare eroarea sistem ORA-02292

```

```

SET VERIFY OFF
DEFINE p_nume = Sales

```

```

DECLARE
  ang_exista EXCEPTION;
  PRAGMA EXCEPTION_INIT(ang_exista,-2292);
BEGIN
  DELETE FROM dep WHERE department_name = '&p_ume';
  COMMIT;
EXCEPTION
  WHEN ang_exista THEN
    DBMS_OUTPUT.PUT_LINE ('nu puteti sterge departamentul cu
      numele ' || '&p_ume' || ' deoarece exista angajati care lucreaza in cadrul
      acestuia');
END;
/
SET VERIFY ON

```

#### 4. [Excepții externe]

Să se scrie un bloc *PL/SQL* care afișează numărul departamentelor în care lucrează angajați al căror salariu este mai mare sau mai mic cu 1000 decât o valoare specificată. Să se tipărească un mesaj adecvat, dacă nu există nici un departament care îndeplinește această condiție.

```

VARIABLE g_mesaj VARCHAR2(100)
SET VERIFY OFF
ACCEPT p_val PROMPT 'introduceti valoarea:'
DECLARE
  v_val    emp.salary%TYPE := &p_val;
  v_inf    emp.salary%TYPE := v_val - 1000;
  v_sup    emp.salary %TYPE := v_val + 1000;
  v_numar  NUMBER(7);
  e_nimeni EXCEPTION;
  e_mai_mult EXCEPTION;
BEGIN
  SELECT COUNT(DISTINCT employee_id)
  INTO   v_numar
  FROM   emp
  WHERE  salary BETWEEN v_inf AND v_sup;
  IF v_numar = 0 THEN
    RAISE e_nimeni;
  ELSIF v_numar > 0 THEN
    RAISE e_mai_mult;
  END IF;
EXCEPTION
  WHEN e_nimeni THEN
    :g_mesaj:='nu exista nici un departament in care sunt angajati
      cu salariul cuprins intre '||v_inf ||' si '||v_sup;
  WHEN e_mai_mult THEN
    :g_mesaj:='exista '||v_numar||' departamente cu angajati avand salariul cuprins intre
      '||v_inf||' si '||v_sup;
  WHEN OTHERS THEN
    :g_mesaj:='au aparut alte erori';
END;
/

SET VERIFY ON
PRINT g_mesaj

```

**[Procedura RAISE\_APPLICATION\_ERROR]**

**5. a)** Să se șteargă salariații asigurați unui cod de departament inexistent în tabelul *departments*. Dacă nu există nici un angajat care a îndeplinit această condiție, să se lanseze o excepție cu mesajul „nici un angajat nu lucreaza in departament inexistent”.

**b)** Să se șteargă angajații al căror comision reprezintă mai mult decât jumătatea diferenței de salariu dintre șeful angajatului respectiv și angajat. Dacă nu există nici un angajat care a îndeplinit această condiție, să se lanseze o excepție cu mesajul „nici un angajat cu comisionul specificat”.

Invocați procedura RAISE\_APPLICATION\_ERROR în secțiuni diferite ale blocului PL/SQL.

```

DECLARE
  e_comision EXCEPTION;
  PRAGMA EXCEPTION_INIT (e_comision, -20777);
BEGIN
  DELETE FROM emp e          -- punctul b)
  WHERE salary * commission_pct > ((SELECT salary
                                     FROM emp
                                     WHERE employee_id = e.manager_id) - salary)/2;

  IF SQL%NOTFOUND THEN
    RAISE e_comision;
  END IF;

  DELETE FROM emp          -- punctul a)
  WHERE department_id NOT IN ( .... ); -- completati subcererea corespunzătoare
                                     --ce problema poate aparea la utilizarea lui
                                     -- NOT IN ?

  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20777, 'nici o stergere'); --apel in sectiunea executabila
  END IF;

EXCEPTION
  WHEN e_comision THEN
    RAISE_APPLICATION_ERROR(-20777, ' nici un angajat cu codul specificat '); -- apel in
    --sectiunea EXCEPTION

END;
/

```

**6.** Să se implementeze un declanșator care nu permite introducerea de salariați în tabelul emp având salariul mai mic decât 1000.

Să se scrie un program care detectează și tratează eroarea „ridicată” de trigger.

```

CREATE OR REPLACE TRIGGER sal_mini
BEFORE INSERT ON emp
FOR EACH ROW
BEGIN
  IF :NEW.salary < 1000 THEN
    RAISE_APPLICATION_ERROR
      (-20005,'angajatii trebuie sa aiba salariul mai mare de 1000');
  END IF;
END;

```

**Obs:** RAISE\_APPLICATION\_ERROR facilitează comunicația dintre *client* și *server*, transmițând aplicației *client* erori specifice aplicației de pe *server* (de obicei, un declanșator). Prin urmare, procedura este doar un mecanism folosit pentru comunicația *server* → *client* a unei erori definite de utilizator, care permite ca procesul *client* să trateze excepția.

Pe stația *client* poate fi scris un program care detectează și tratează eroarea.

```

DECLARE

```

```

--declarare exceptie
nu_accepta EXCEPTION;
--asociază nume, codului eroare folosit in trigger
PRAGMA EXCEPTION_INIT(nu_accepta,-20005);
-- variabila PL/SQL in care retinem codul maxim + 1
v_cod emp.employee_id%TYPE;
BEGIN
SELECT MAX(employee_id) + 1 INTO v_cod
FROM emp;
-- incercare inserare
INSERT INTO emp (employee_id, last_name, email, hire_date, salary)
VALUES (v_cod, 'someone', 'smth@smth.com', SYSDATE, 900);
EXCEPTION
/* tratare exceptie */
WHEN nu_accepta THEN
DBMS_OUTPUT.PUT_LINE('trigger-ul lanseaza o eroare cu mesajul' || SQLERRM );
/* SQLERRM va returna mesaj din RAISE_APPLICATION_ERROR */
END;

```

### [Cazuri speciale în tratarea excepțiilor]

7. Să se scrie un program (bloc) PL/SQL care să determine codul și salariul angajatului având salariul minim în departamentul cerut de utilizator și apoi să se introducă informațiile găsite în tabelul mesaje.

**Obs:** Dacă se declanșează o excepție într-un bloc simplu, atunci se face saltul la partea de tratare (*handler*) a acesteia, iar după ce este terminată tratarea erorii se iese din bloc (instrucțiunea *END*).

Comentați corectitudinea următoarei soluții propuse:

```

ACCEPT p_dep PROMPT 'Introduceți un cod de departament '
DECLARE
v_cod emp.employee_id%TYPE;
v_sal emp.salary%TYPE;
v_dep emp.department_id%TYPE := &p_dep;
BEGIN
DELETE from mesaje;
v_cod := -1;
v_sal := 0;
SELECT employee_id, salary INTO v_cod, v_sal
FROM emp e
WHERE salary = (SELECT MIN(salary)
FROM emp
WHERE department_id = e.department_id);
--poate declansa exceptia NO_DATA_FOUND sau TOO_MANY_ROWS
--se ajunge la comanda urmatoare?
INSERT INTO mesaje (v_cod || v_sal);
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20500, 'Nici o linie gasita') ;
WHEN TOO_MANY_ROWS THEN
RAISE_APPLICATION_ERROR(-20501, 'Prea multi angajati') ;
END;

```

Deficiența anterioară (iesirea din bloc inaintea se poate rezolva incluzând într-un subbloc comanda *SELECT* care a declanșat excepția.

```

BEGIN
DELETE from mesaje;
v_cod := -1;

```

```

v_sal := 0;
<<subbloc>>
BEGIN
    SELECT employee_id, salary INTO v_cod, v_sal
    FROM emp e
    WHERE salary = (SELECT MIN(salary)
                    FROM emp
                    WHERE department_id = e.department_id);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20500, 'Nici o linie gasita');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20501, 'Prea multi angajati');
    /* dupa ce se trateaza exceptiile, controlul este
       transferat blocului de nivel superior, de fapt
       comenzii INSERT */
END <<subbloc>>;
INSERT INTO mesaje (v_cod || v_sal);
END;

```

**8.** Să se creeze un bloc PL/SQL care determină:

- numele, salariul si vechimea angajatului având salariul maxim în departamentul în care salariul mediu este minim;
- codul și data angajării celui mai bine platit angajat din Oxford.
- numele și salariul angajatului având cea mai mica vechime.

Dacă vreuna dintre comenzi lansează excepția TOO\_MANY\_ROWS, să se introducă în tabelul *mesaje* informații despre comanda care a lansat această excepție.

*Solutia 1:* Introducerea unui contor care să identifice instrucțiunea SQL.

```

DECLARE
v_sel_cont NUMBER(2):=1;
v_ume employees.last_name%TYPE;
v_sal employees.salary%TYPE;
v_vechi NUMBER(2);
v_cod employees.employee_id%TYPE;
v_data employees.hire_date%TYPE;
BEGIN
SELECT last_name, salary, ROUND((sysdate-hire_date)/365)
INTO v_ume, v_sal, v_vechi
FROM employees e
WHERE department_id IN (SELECT department_id
                        FROM employees
                        GROUP BY department_id
                        HAVING AVG(salary) = (SELECT MIN(AVG(salary))
                                             FROM employees
                                             GROUP BY department_id))

AND salary = (SELECT MAX(salary)
              FROM employees
              WHERE e.department_id = department_id);

v_sel_cont:=2;
SELECT employee_id, hire_date
INTO v_cod, v_data
FROM employees e, departments d, locations l
WHERE e.department_id = d.department_id AND d.department_id = l.location_id

```



```

AND (city, salary) IN (SELECT city, MAX(salary)
                       FROM employees e, departments d, locations l
                       WHERE e.department_id = d.department_id
                             AND d.department_id = l.location_id
                             AND INITCAP(city) = 'Oxford');

```

```

v_sel_cont:=3;
SELECT last_name, salary
INTO v_nume, v_sal
FROM employees
WHERE hire_date = (SELECT MAX(hire_date)
                   FROM employees);

```

```

EXCEPTION
WHEN TOO_MANY_ROWS THEN
  INSERT INTO mesaje(rezultat)
  VALUES ('comanda SELECT ' || TO_CHAR(v_sel_cont) ||
          ' gaseste prea multe date');
END;

```

*Solutia 2:* Introducerea fiecărei instrucțiuni SQL într-un subbloc.

```

BEGIN
  BEGIN
    SELECT ...
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      INSERT INTO mesaje(rezultat)
      VALUES('SELECT 1 gaseste prea multe date');
  END;
  BEGIN
    SELECT ...
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      INSERT INTO mesaje(rezultat)
      VALUES('SELECT 2 gaseste prea multe date');
  END;
  ...
END;

```

Ce deosebire este între cele două variante? (care este mai avantajoasă?)

### [Propagarea excepțiilor]

9. Să se rețină în tabelul mesaje raportul dintre salariu și comision pentru angajatul având cea mai mare vechime.

*Variante:*

<pre> DECLARE   v_var NUMBER(10,3); BEGIN   SELECT salary/comission_pct   INTO v_var   FROM emp   WHERE hire_date = (SELECT   MIN(hire_date) FROM emp);   &lt;&lt;eticheta&gt;&gt;   INSERT INTO mesaje VALUES   (v_var); </pre>	<pre> DECLARE   v_var NUMBER(10,3); BEGIN   IF nvl(commission_pct, 0)=0 THEN     GOTO eticheta; --este posibil?   END IF;   SELECT salary/comission_pct   INTO v_var   FROM emp   WHERE hire_date = (SELECT   MIN(hire_date) FROM emp); </pre>
--	--

<pre>EXCEPTION WHEN ZERO_DIVIDE THEN v_var:=0; GOTO &lt;&lt;eticheta&gt;&gt;; --este posibil? END;</pre>	<pre>INSERT INTO mesaje VALUES (v_var); EXCEPTION &lt;&lt;eticheta&gt;&gt; WHEN ZERO_DIVIDE THEN v_var:=0; END;</pre>
--	---

**Obs:** Instrucțiunea *GOTO* nu permite:

- saltul la secțiunea de tratare a unei excepții;
- saltul de la secțiunea de tratare a unei excepții, în blocul curent.

Comanda *GOTO* permite totuși saltul de la secțiunea de tratare a unei excepții la un bloc care include blocul curent.

Cum se poate remedia situația de mai sus? Schimbați blocul propus (utilizând un subbloc) astfel încât să funcționeze corespunzător.

### 10. [Excepție sesizată în secțiunea executabilă a unui subbloc, dar netratată în subbloc]

Să se declare un bloc în care se va crea un subbloc ce lansează o excepție e2. Subblocul nu va conține handler pentru e2, în schimb un astfel de handler se va afla în bloc. Ce se întâmplă la execuția blocului?

**Obs:** Excepția este sesizată în subbloc, dar nu este tratată în acesta și atunci se propagă spre blocul exterior. Regula poate fi aplicată de mai multe ori.

```
DECLARE
e1 EXCEPTION;
e2 EXCEPTION;
BEGIN
<<subbloc>>
BEGIN
RAISE e2; --exceptia e2 sesizata in subbloc
EXCEPTION
WHEN e1 THEN
RAISE_APPLICATION_ERROR(-20200, 'Mesaj pentru e1 in subbloc');
--exceptia e2 nu este tratata in subbloc
END <<subbloc>>;
EXCEPTION
WHEN e2 THEN
RAISE_APPLICATION_ERROR(-20202, 'Mesaj pentru e2 in bloc');
/* exceptia e2 s-a propagat spre blocul exterior unde a
fost tratata, apoi controlul trece in exteriorul blocului */
END;
```

### 11. [Excepție sesizată în secțiunea declarativă]

Să se realizeze un program care calculează numărul departamentelor în care lucrează angajați, exemplificând erorile care pot apărea în secțiunea declarativă.

```
BEGIN
DECLARE
nr_dep NUMBER(3) := 'XYZ'; --eroare
BEGIN
SELECT COUNT (DISTINCT department_id)
INTO nr_dep
FROM emp;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Eroare bloc intern:' || SQLERRM);
```

```

END;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare bloc extern:' || SQLERRM );
END;

```

**Obs:** Dacă în secțiunea declarativă este generată o excepție, atunci aceasta se propagă către blocul exterior, unde are loc tratarea acesteia.

## 12. [Excepție sesizată în secțiunea *EXCEPTION*]

Să se declare un bloc PL/SQL, care conține un subbloc ce lansează o excepție e1 care la rândul ei lansează o altă excepție e2. Excepția e2 va avea câte un handler atât în bloc, cât și în subbloc. Unde va fi tratată e2 în urma execuției blocului?

**Obs:** Dacă excepția este sesizată în secțiunea *EXCEPTION*, ea se propagă imediat spre blocul exterior.

```

DECLARE
  e1 EXCEPTION;
  e2 EXCEPTION;
BEGIN
  <<subbloc>>
  BEGIN
    RAISE e1; --sesizare exceptie e1
  EXCEPTION
    WHEN e1 THEN
      RAISE e2; --sesizare exceptie e2
    WHEN e2 THEN
      RAISE_APPLICATION_ERROR(-20200, 'Handler in subbloc');
      /* exceptia este propagata spre blocul exterior
      cu toate ca exista aici un handler pentru ea */
  END <<subbloc>>;
EXCEPTION
  WHEN e2 THEN
    RAISE_APPLICATION_ERROR(-20202, 'Handler in bloc');
  --exceptia e2 este tratata in blocul exterior
END;

```

13. Compilați una din procedurile de la laboratoarele precedente și afișați erorile de compilare.

```

SELECT LINE, POSITION, TEXT
FROM USER_ERRORS
WHERE UPPER(NAME) = UPPER('nume_procedura');

```