Laborator 7 PL/SQL

Tratarea erorilor
Exerciții:
- Excepții interne predefinite
- Excepții interne nepredefinite
- Excepții externe
- Procedura RAISE_APPLICATION_ERROR
- Cazuri speciale în tratarea excepțiilor
- Propagarea excepțiilor
- Excepție sesizată în secțiunea executabilă a unui subbloc, dar netratată în subbloc
- Excepție sesizată în secțiunea declarativă

Tratarea erorilor

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

```
EXCEPTION
   WHEN nume_exceptie1 [OR nume_exceptie2 ...] THEN
       secventa_de_instructiuni_1;
   [WHEN nume_exceptie3 [OR nume_exceptie4 ...] THEN
       secventa_de_instructiuni_2;] ...
   [WHEN OTHERS THEN secventa_de_instructiuni_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 excepii);
- 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 seciunea declarativ și sunt tratate implicit de tre server-ul Oracle. Ele sunt referite prin nume (CURSOR_ALREADY_OPEN, DUP_VAL_ON_INDEX, NO_DATA_FOUND, TOO_MANY_ROWS, ZERO_DIVIDE)
    - **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 directiva de compilare PRAGMA EXCEPTION_INIT [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_exceptie EXCEPTION; -- declarare exceptie
BEGIN
   ...
   RAISE nume_exceptie; -- declansare exceptie
   -- codul care urmeaza nu mai este executat
   ...
EXCEPTION
   WHEN nume_exceptie 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_exceptie, 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 depus în USER_SOURCE). Dacă nu sunt erori, apare mesajul NO ROWS SELECTED.

Exerciții:

1. Creați tabelul erori_pnu 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_pnu.
Varianta 1 (captarea erorii interne a sistemului)

```
CREATE TABLE erori_pnu (cod NUMBER, mesaj VARCHAR2(100));
SET VERIFY OFF
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_pnu%ROWTYPE;
BEGIN
    SELECT *
    INTO v_emp
    FROM emp_pnu
    WHERE department_id = v_dep
    AND hire_date = (SELECT MAX(hire_date)
                     FROM emp_pnu
                     WHERE department_id = v_dep);
    DBMS_OUTPUT.PUT_LINE(v_emp.last_name || ' ' ||v_emp.salary);
    INSERT INTO erori_pnu
        VALUES (0, 'Fara eroare pt departamentul '|| v_dep);
EXCEPTION
    WHEN OTHERS THEN
        eroare_cod := SQLCODE;
        eroare_mesaj := SUBSTR(SQLERRM,1,100);
        INSERT INTO erori_pnu
            VALUES (eroare_cod, eroare_mesaj);
END;
/
SELECT *
FROM erori_pnu;
```

Varianta 2 (definirea unei excepții de către utilizator)

```
SET SERVEROUTPUT ON
UNDEFINE p_cod
DECLARE
    eroare_cod NUMBER;
    eroare_mesaj VARCHAR2(100);
    v_dep departments.department_id%TYPE := &p_cod;
    v_emp emp_pnu%ROWTYPE;
    v_num NUMBER;
    exceptie EXCEPTION;
BEGIN
    SELECT COUNT(*)
    INTO v_num --(1)
    FROM emp_pnu
    WHERE department_id = v_dep
    AND hire_date = (SELECT MAX(hire_date )
                     FROM emp_pnu
                     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_pnu
    WHERE department_id = v_dep
    AND hire_date = (SELECT MAX(hire_date )
                     FROM emp_pnu
                     WHERE department_id = v_dep);
    /* Daca nu aveam instructiunile (1) si (2),
    s-ar fi ridicat exceptia cu urmatorul IF? */
    /*
    IF SQL%ROWCOUNT != 1 THEN
       -- fara (1) si (2) nu se ajunge la aceasta instructiune,
       -- daca (3) declanseaza 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);
    INSERT INTO erori_pnu
        VALUES (0, 'Fara eroare pt departamentul '|| v_dep);

EXCEPTION
    WHEN exceptie THEN
        eroare_cod := -20200;
        eroare_mesaj := 'prea multe sau prea putine linii';
        INSERT INTO erori_pnu
            VALUES (eroare_cod, eroare_mesaj);
```

```
END;
/
SELECT *
FROM erori_pnu;
```

Excepții interne predefinite

2. Creați tabelul mesaje_pnu 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.

a) Dacă rezultatul interogării returnează mai mult decât o linie, atunci să se trateze excepția și să se insereze în tabelul mesaje_pnu textul „Mai multe departamente“.

b) Dacă rezultatul interogării nu returnează nici o linie, atunci să se trateze excepția și să se insereze în tabelul mesaje_pnu textul „Nici un departament“.

c) Dacă rezultatul interogării este o singură linie, atunci să se insereze în tabelul mesaje_pnu numele departamentului și managerul acestuia.

d) Să se trateze orice altă eroare, inserând în tabelul mesaje_pnu textul „Au apărut alte erori“.

```
CREATE TABLE mesaje_pnu (rezultate VARCHAR2(50));
UNDEFINE p_locatie
ACCEPT p_locatie PROMPT 'Introduceti locatia:'
DECLARE
    v_nume_dep dept_pnu.department_name%TYPE;
    v_manager dept_pnu.manager_id%TYPE;
    v_locatie dept_pnu.location_id%TYPE:='&p_locatie';
BEGIN
    SELECT department_name, manager_id
    INTO v_nume_dep, v_manager
    FROM dept_pnu
    WHERE location_id = v_locatie;
    INSERT INTO mesaje_pnu (rezultate)
        VALUES (v_nume_dep||'-'||v_manager);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        INSERT INTO mesaje_pnu (rezultate)
        VALUES ('Nici un departament.');
    WHEN TOO_MANY_ROWS THEN
        INSERT INTO mesaje_pnu (rezultate)
        VALUES ('Mai multe departamente.');
    WHEN OTHERS THEN
        INSERT INTO mesaje_pnu (rezultate)
        VALUES ('Au aparut alte erori.');
END;
/
```

```
SELECT *
FROM mesaje_pnu;
```

Excepții interne nepredefinite

3. 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 dept_pnu
    ADD CONSTRAINT dept_pnu_pk PRIMARY KEY (department_id);
ALTER TABLE emp_pnu
    ADD CONSTRAINT emp_pnu_pk PRIMARY KEY (employee_id);
ALTER TABLE emp_pnu
    ADD CONSTRAINT emp_dept_pnu_fk FOREIGN KEY(department_id)
        REFERENCES dept_pnu(department_id);
-- ALTER TABLE dept_pnu DROP CONSTRAINT dept_pnu_pk;
-- ALTER TABLE emp_pnu DROP CONSTRAINT emp_pnu_pk;
-- ALTER TABLE emp_pnu DROP CONSTRAINT emp_dept_pnu_fk;

-- Ce se întâmpla daca se încearca stergerea unui departament în care lucreaza
angajati?
DELETE FROM dept_pnu
WHERE department_id = 30;
--apare eroarea sistem ORA-02292
DEFINE p_nume = 'Sales'
DECLARE
ang_exista EXCEPTION;
PRAGMA EXCEPTION_INIT(ang_exista,-2292);
BEGIN
    DELETE FROM dept_pnu
    WHERE department_name = '&p_nume';
    COMMIT;
EXCEPTION
    WHEN ang_exista THEN
        DBMS_OUTPUT.PUT_LINE ('nu puteti sterge departamentul cu numele ' ||
'&p_nume' || ' deoarece exista angajati care lucreaza in cadrul acestuia');
END;
/
```

Excepții externe

4. 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 tiprească un mesaj adecvat, dacă nu există nici un departament care îndeplinește această condiție.

```
VARIABLE g_mesaj VARCHAR2(100)
ACCEPT p_val PROMPT 'introduceti valoarea:'
```

```
DECLARE
    v_val emp_pnu.salary%TYPE := &p_val;
    v_inf emp_pnu.salary%TYPE := v_val - 1000;
    v_sup emp_pnu.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_pnu
    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;
/
PRINT g_mesaj
```

Procedura RAISE_APPLICATION_ERROR

5. a) Să se șteargă salariații asignați 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.

```
ALTER TABLE emp_pnu DISABLE ALL TRIGGERS;
DELETE FROM emp_pnu;
DELETE FROM dept_pnu;
INSERT INTO dept_pnu
SELECT *
FROM departments;
INSERT INTO emp_pnu
SELECT *
```

```
FROM employees;
SELECT * FROM emp_pnu;
SELECT * FROM dept_pnu;
-- punctul a
SELECT *
FROM emp_pnu
WHERE department_id NOT IN (SELECT department_id
                            FROM dept_pnu);
-- Ce problema poate aparea la utilizarea lui NOT IN ?
DELETE FROM emp_pnu
WHERE department_id NOT IN (SELECT department_id
                            FROM dept_pnu);
BEGIN
    DELETE FROM emp_pnu
    WHERE NVL(department_id,0) NOT IN (SELECT department_id
                                FROM dept_pnu);
    IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR(-20777, 'nici un angajat nu lucreaza in
departament inexistent'); --apel in sectiunea executabila
    ELSE
        DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' linii sterse');
    END IF;
    COMMIT;
END;
/
-- punctul b
SELECT *
FROM emp_pnu e
WHERE salary * NVL(commission_pct,0) > ((SELECT salary
                                         FROM emp_pnu
                                         WHERE employee_id =
NVL(e.manager_id,0)) - e.salary)/2;

DECLARE
    e_comision EXCEPTION;
    PRAGMA EXCEPTION_INIT (e_comision, -20778);
BEGIN
    DELETE FROM emp_pnu e
    WHERE salary * NVL(commission_pct,0) > ((SELECT salary
                                             FROM emp_pnu
                                             WHERE employee_id =
NVL(e.manager_id,0)) - e.salary)/2;
    IF SQL%NOTFOUND THEN
        RAISE e_comision;
    ELSE
        DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' linii sterse');
    END IF;
    COMMIT;
EXCEPTION
    WHEN e_comision THEN
        RAISE_APPLICATION_ERROR(-20778, 'nici un angajat cu comisionul
```

```
specificat'); -- apel in sectiunea EXCEPTION
END;
/
```

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

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

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.

```
CREATE OR REPLACE TRIGGER sal_mini_pnu
    BEFORE INSERT ON emp_pnu
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;
/
DECLARE
    --declarare exceptie
    nu_accepta EXCEPTION;
    --asociaza nume codului eroare folosit in trigger
    PRAGMA EXCEPTION_INIT(nu_accepta,-20005);
    -- variabila PL/SQL in care retinem codul maxim + 1
    v_cod_max emp_pnu.employee_id%TYPE;
BEGIN
    SELECT MAX(employee_id) + 1
    INTO v_cod_max
    FROM emp_pnu;
    -- incercare inserare
    INSERT INTO emp_pnu (employee_id, last_name, email, hire_date, salary)
    VALUES (v_cod_max, 'someone', 'smth@g.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_pnu.

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.

```
SELECT department_id, COUNT(employee_id)
FROM emp_pnu
WHERE (department_id,salary) IN (SELECT department_id, MIN(salary)
                                 FROM emp_pnu
                                 GROUP BY department_id
                                 )
GROUP BY department_id;
-- sau
SELECT department_id, COUNT(employee_id)
FROM emp_pnu e
WHERE salary = (SELECT MIN(salary)
                FROM emp_pnu
                WHERE department_id = e.department_id
                )
GROUP BY department_id;

UNDEFINE p_dep
ACCEPT p_dep PROMPT 'Introduceti un cod de departament '
DECLARE
    v_cod emp_pnu.employee_id%TYPE;
    v_sal emp_pnu.salary%TYPE;
    v_dep emp_pnu.department_id%TYPE := &p_dep;
BEGIN
    DELETE FROM mesaje_pnu;
    v_cod := -1;
    v_sal := 0;
    SELECT employee_id, salary
    INTO v_cod, v_sal
    FROM emp_pnu e
    WHERE salary = (SELECT MIN(salary)
                    FROM emp_pnu
                    WHERE department_id = e.department_id) AND
        e.department_id = v_dep;
    --poate declansa exceptia NO_DATA_FOUND sau TOO_MANY_ROWS
    --se ajunge la comanda urmatoare?
    INSERT INTO mesaje_pnu
    VALUES ('angajatul cu codul '|| v_cod || ' are salariul ' || 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;
/
SELECT *
FROM mesaje_pnu;
```

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

```
UNDEFINE p_dep
ACCEPT p_dep PROMPT 'Introduceti un cod de departament '
DECLARE
    v_cod emp_pnu.employee_id%TYPE;
    v_sal emp_pnu.salary%TYPE;
    v_dep emp_pnu.department_id%TYPE := &p_dep;
BEGIN
    DELETE FROM mesaje_pnu;
    v_cod := -1;
    v_sal := 0;
    <<subbloc>>
    BEGIN
        SELECT employee_id, salary
        INTO v_cod, v_sal
        FROM emp_pnu e
        WHERE salary = (SELECT MIN(salary)
                        FROM emp_pnu
                        WHERE department_id = e.department_id) AND
            e.department_id = v_dep;
    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_pnu
    VALUES ('angajatul cu codul '|| v_cod || ' are salariul ' || v_sal);
END;
/
SELECT *
FROM mesaje_pnu;
```

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 plătit angajat din Oxford.
- numele și salariul angajatului având cea mai mică vechime.

Dacă vreuna dintre comenzi lansează excepția TOO_MANY_ROWS, să se introducă în tabelul mesaje_pnu informații despre comanda care a lansat aceasta exceptie.

Soluția 1: Introducerea unui contor care să identifice instrucțiunea SQL.

```
DECLARE
    v_contor NUMBER(2):=1;
    v_nume employees.last_name%TYPE;
    v_salariu employees.salary%TYPE;
    v_vechime NUMBER(2);
    v_cod employees.employee_id%TYPE;
    v_data_ang employees.hire_date%TYPE;
BEGIN
    SELECT last_name, salary, ROUND((sysdate-hire_date)/365)
    INTO v_nume, v_salariu, v_vechime
    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_contor:=2;
    SELECT employee_id, hire_date
    INTO v_cod, v_data_ang
    FROM employees e, departments d, locations l
    WHERE e.department_id = d.department_id AND
          d.location_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.location_id = l.location_id AND
                                   INITCAP(city) = 'Oxford'
                             GROUP BY city);
    v_contor:=3;
    SELECT last_name, salary
    INTO v_nume, v_salariu
    FROM employees
    WHERE hire_date = (SELECT MAX(hire_date)
                       FROM employees);
    EXCEPTION
        WHEN TOO_MANY_ROWS THEN
            INSERT INTO mesaje_pnu (rezultate)
                VALUES ('comanda SELECT ' || TO_CHAR(v_contor) || ' gaseste
prea multe date');
END;
```

```
/
SELECT *
FROM mesaje_pnu;
```

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

```
BEGIN
    BEGIN
        SELECT ...
    EXCEPTION
        WHEN TOO_MANY_ROWS THEN
            INSERT INTO mesaje_pnu(rezultate)
                VALUES('SELECT 1 gaseste prea multe date');
    END;
    BEGIN
        SELECT ...
    EXCEPTION
        WHEN TOO_MANY_ROWS THEN
            INSERT INTO mesaje_pnu(rezultate)
                VALUES('SELECT 2 gaseste prea multe date');
    END;
    BEGIN
        SELECT ...
    EXCEPTION
        WHEN TOO_MANY_ROWS THEN
            INSERT INTO mesaje_pnu(rezultate)
                VALUES('SELECT 3 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_pnu raportul dintre salariu și comision pentru angajatul având cea mai mare vechime.

```
DECLARE
    v_var NUMBER(10,3);
BEGIN
    SELECT salary/commission_pct
    INTO v_var
    FROM emp_pnu
    WHERE hire_date = (SELECT
                        MIN(hire_date)
                       FROM emp_pnu);
    <<eticheta>>
    INSERT INTO mesaje_pnu
        VALUES (v_var);
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        v_var:=0;
        --GOTO <<eticheta>>; --este
posibil?
END;
```

```
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_pnu
    WHERE hire_date = (SELECT
                         MIN(hire_date)
                        FROM emp_pnu);
    INSERT INTO mesaje_pnu
        VALUES (v_var);
EXCEPTION
<<eticheta>>
    WHEN ZERO_DIVIDE THEN
        v_var:=0;
END;
```

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.

Excepție sesizată în secțiunea executabilă a unui subbloc, dar netratată în subbloc

10. 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;
```

Excepție sesizată în secțiunea declarativă

11. 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_pnu;
    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.

Excepție sesizată în secțiunea EXCEPTION

12. 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');
```