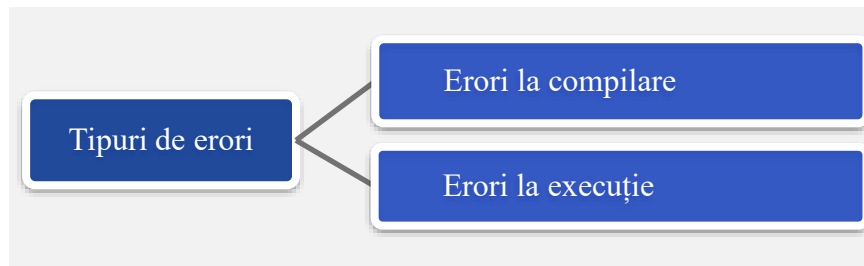


## CUPRINS

9. <i>PL/SQL</i> – Gestiunea excepțiilor .....	2
9.1. Secțiunea de tratare a excepțiilor .....	4
9.2. Funcții pentru identificarea excepțiilor .....	5
9.3. Excepții interne predefinite .....	5
9.4. Excepții interne nepredefinite .....	7
9.5. Excepții externe .....	8
9.6. Cazuri speciale în tratarea excepțiilor .....	11
9.7. Activarea excepțiilor .....	13
9.8. Propagarea excepțiilor .....	14
9.8.1 Excepție declanșată în secțiunea executabilă .....	15
9.8.2 Excepție declanșată în secțiunea declarativă .....	15
9.8.3 Excepție declanșată în secțiunea <i>EXCEPTION</i> .....	16
9.9. Informații despre erori .....	17
Bibliografie .....	18

## 9. PL/SQL – Gestiunea excepțiilor

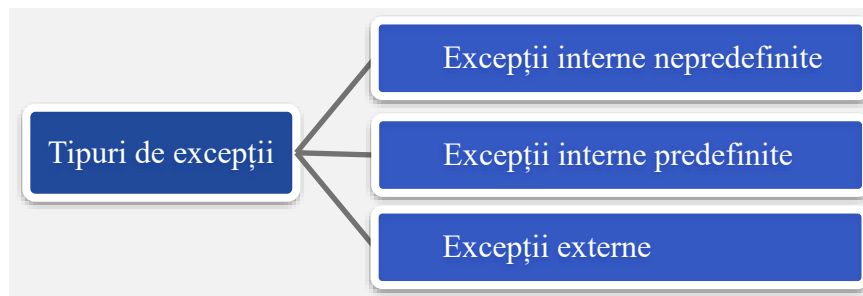
- Mecanismul de gestiune a excepțiilor permite utilizatorului să definească și să controleze comportamentul programului atunci când acesta generează o eroare.



**Fig. 9.1.** Tipuri de erori

- Într-un program *PL/SQL* pot să apară:
  - erori la compilare
    - sunt detectate de motorul *PL/SQL*;
    - programul nu poate trata aceste erori deoarece nu a fost încă executat;
    - programatorul trebuie să corecteze erorile și să execute din nou programul;
  - erori la execuție
    - sunt denumite excepții;
    - pot apărea datorită deficiențelor de proiectare, defectărilor la nivel *hardware*, greșelilor de cod etc.;
    - în program trebuie prevăzută apariția unei astfel de erori și specificat modul concret de tratare a acesteia;
      - atunci când apare eroarea este declanșată o excepție, iar controlul trece la secțiunea de tratare a excepțiilor, unde va avea loc tratarea erorii;
      - dacă excepția nu este tratată, atunci aceasta se va propaga în mediul din care a fost lansat programul;
      - nu pot fi anticipate toate excepțiile posibile, dar prin mecanismul de tratare a excepțiilor se poate permite programului să își continue execuția și în prezența anumitor erori.
- Excepțiile pot fi definite și tratate la nivelul fiecărui bloc din program (bloc principal, funcții și proceduri, blocuri interioare acestora).
  - Execuția unui bloc se termină întotdeauna atunci când apare o excepție, dar se pot executa acțiuni ulterioare apariției acesteia, într-o secțiune specială de tratare a excepțiilor.

- Posibilitatea de a da nume fiecărei excepții, de a izola tratarea erorilor într-o secțiune particulară, de a declanșa automat erori (în cazul excepțiilor interne) îmbunătățește lizibilitatea și fiabilitatea programului. Prin utilizarea excepțiilor și a rutinelor de tratare a excepțiilor, un program *PL/SQL* devine robust și capabil să trateze atât erorile așteptate, cât și cele neașteptate ce pot apărea în timpul execuției.



**Fig. 9.2.** Tipuri de excepții

- Există următoarele tipuri de excepții:
  - excepții interne nepredefinite
    - au un cod de eroare, dar nu au nume asociat decât dacă acesta este precizat de către utilizator;
  - excepții interne predefinite
    - sunt excepții interne care au nume asociat de către sistem;
  - excepții externe
    - sunt excepții definite de utilizator în blocuri *PL/SQL* anonime, subprograme sau pachete.
- Excepțiile interne:
  - se produc atunci când un bloc *PL/SQL* nu respectă o regulă *Oracle* sau depășește o limită a sistemului de operare;
  - pot fi independente de structura bazei de date sau pot să apară datorită nerespectării constrângerilor statice implementate în structură (*PRIMARY KEY*, *FOREIGN KEY*, *NOT NULL*, *UNIQUE*, *CHECK*);
- Atunci când apare o eroare *Oracle*, excepția asociată acesteia se declanșează implicit.
  - De exemplu, dacă apare eroarea *ORA-01403* (deoarece o comandă *SELECT* nu întoarce nicio linie) atunci implicit *PL/SQL* activează excepția *NO\_DATA\_FOUND* (al cărui cod este 100).
  - Cu toate că fiecare astfel de excepție are asociat un cod specific, ele trebuie referite prin nume.

	Excepții interne nepredefinite	Excepții interne predefinite	Excepții externe
<b>Proprietar</b>	Sistemul	Sistemul	Utilizatorul
<b>Cod asociat</b>	Da	Da	Doar dacă utilizatorul îl atribuie
<b>Nume asociat</b>	Doar dacă utilizatorul îl atribuie	Da	Da
<b>Declanșare automată</b>	Da	Da	Nu
<b>Declanșare explicită</b>	Opțional	Opțional	Da

Fig. 9.3. Caracteristicile excepțiilor

## 9.1. Secțiunea de tratare a excepțiilor

- Tratarea excepțiilor se realizează în secțiunea *EXCEPTION* a unui bloc *PL/SQL*.
- Sintaxa:

```

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;
```

- Clauza *WHEN OTHERS* trebuie să fie ultima clauză specificată și trebuie să fie unică.
  - Toate excepțiile care nu au fost specificate explicit vor fi captate prin această clauză.

## 9.2. Funcții pentru identificarea excepțiilor

- Funcția *SQLCODE*:
  - obține codul excepției;
  - întoarce o valoare de tip numeric;
  - codul excepției este:
    - un număr negativ, în cazul unei erori interne;
    - 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.
- Funcția *SQLERRM*:
  - obține mesajul asociat excepției;
  - întoarce un șir de caractere;
  - lungimea maximă a mesajului este de 512 caractere;
  - mesajul asociat excepției declanșate poate fi furnizat și de funcția *DBMS\_UTILITY.FORMAT\_ERROR\_STACK*.

**Exemplul 9.1** – **vezi curs**

## 9.3. Excepții interne predefinite

- Excepțiile interne predefinite (erori de tip *ORA-n*):
  - nu trebuie declarate în secțiunea declarativă a blocului *PL/SQL*;
  - sunt declanșate implicit de către *server-ul Oracle*;
  - sunt referite prin numele asociat lor;
    - *PL/SQL* declară aceste excepții în pachetul *STANDARD*.

Nume excepție	Cod	Descriere
ACCESS_INTO_NULL	-6530	Asignare de valori atributelor unui obiect neinițializat.
CASE_NOT_FOUND	-6592	Nu este selectată nici una din clauzele <i>WHEN</i> ale lui <i>CASE</i> și nu există nici clauza <i>ELSE</i> .
COLLECTION_IS_NULL	-6531	Aplicarea unei metode (diferite de <i>EXISTS</i> ) unui tabel imbricat sau unui vector neinițializat.

CURSOR_ALREADY_OPEN	-6511	Deschiderea unui cursor care este deja deschis.
DUP_VAL_ON_INDEX	-1	Detectarea unei dubluri într-o coloană unde acestea sunt interzise.
INVALID_CURSOR	-1001	Operație ilegală asupra unui cursor.
INVALID_NUMBER	-1722	Conversie nepermisă de la tipul șir de caractere la număr.
LOGIN_DENIED	-1017	Nume sau parolă incorecte.
NO_DATA_FOUND	+100	Comanda <i>SELECT</i> nu întoarce nicio linie.
NOT_LOGGED_ON	-1012	Programul <i>PL/SQL</i> apelează baza de date fără să fie conectat la <i>Oracle</i> .
PROGRAM_ERROR	-6501	<i>PL/SQL</i> are o problemă internă.
ROWTYPE_MISMATCH	-6504	Incompatibilitate între parametrii actuali și formali, la deschiderea unui cursor parametrizat.
SELF_IS_NULL	-30625	Apelul unei metode când instanța este <i>NULL</i> .
STORAGE_ERROR	-6500	<i>PL/SQL</i> are probleme cu spațiul de memorie.
SUBSCRIPT_BEYOND_COUNT	-6533	Referire la o componentă a unui tablou imbricat sau vector, folosind un index mai mare decât numărul elementelor colecției respective.
SUBSCRIPT_OUTSIDE_LIMIT	-6532	Referire la o componentă a unui tabel imbricat sau vector, folosind un index care este în afara domeniului (de exemplu, -1).
SYS_INVALID_ROWID	-1410	Conversia unui șir de caractere într-un <i>ROWID</i> nu se poate face deoarece șirul nu reprezintă un <i>ROWID</i> valid.
TIMEOUT_ON_RESOURCE	-51	Expirarea timpului de așteptare pentru eliberarea unei resurse.
TRANSACTION_BACKED_OUT	-61	Tranzacția este anulată datorită unei interblocări.
TOO_MANY_ROWS	-1422	Comanda <i>SELECT</i> întoarce mai multe linii.

VALUE_ERROR	-6502	Apariția unor erori în conversii, constrângeri sau erori aritmetice.
ZERO_DIVIDE	-1476	Sesizarea unei împărțiri la zero.



- ❖ Aceeași excepție poate să apară în diferite circumstanțe.
- ❖ De exemplu, excepția *NO\_DATA\_FOUND* poate fi generată fie pentru că o interogare nu întoarce un rezultat, fie pentru că se referă un element al unui tablou *PL/SQL* care nu a fost definit (nu are atribuită o valoare).
- ❖ Dacă într-un bloc *PL/SQL* apar ambele situații, este greu de stabilit care dintre ele a generat eroarea și este necesară restructurarea blocului, astfel încât acesta să poată diferenția cele două situații.

**Exemplul 9.2** – vezi curs



- ❖ Deși excepțiile interne sunt lansate implicit (automat) de către sistem, sunt cazuri în care utilizatorul le poate invoca explicit.

**Exemplul 9.3** – vezi curs

## 9.4. Excepții interne nepredefinite

- Excepțiile interne nepredefinite se declară în secțiunea declarativă a blocului *PL/SQL* și sunt declanșate implicit de către *server-ul Oracle*.
- Diferențierea acestor erori este posibilă doar cu ajutorul codului asociat lor.
- Există două metode de tratare a excepțiilor interne nepredefinite, folosind:
  - clauza *WHEN OTHERS* din secțiunea *EXCEPTION* a blocului;
  - directiva de compilare (pseudo-instrucțiune) *PRAGMA EXCEPTION\_INIT*.
- Directiva *PRAGMA EXCEPTION\_INIT*
  - permite asocierea unui nume pentru o excepție al cărui cod de eroare intern este specificat;
  - având un nume specificat pentru excepție, permite referirea acesteia în secțiunea *EXCEPTION* a blocului;
  - este procesată în momentul compilării (nu la execuție);

- trebuie să apară în partea declarativă a unui bloc, pachet sau subprogram, după definirea numelui excepției;
  - poate să apară de mai multe ori într-un program; de asemenea, pot fi asignate mai multe nume pentru același cod de eroare.
- Pentru a trata o eroare folosind directiva *PRAGMA EXCEPTION\_INIT* trebuie urmați pașii de mai jos:
    - 1) se declară numele excepției în partea declarativă a blocului:  
`nume_excepție EXCEPTION;`
    - 2) se asociază numele excepției cu un cod de eroare standard *Oracle*:  
`PRAGMA EXCEPTION_INIT (nume_excepție, cod_eroare);`
    - 3) se referă excepția în secțiunea *EXCEPTION* a blocului (excepția este tratată automat, fără a fi necesară comanda *RAISE*):  
`WHEN nume_excepție THEN set_de_instrucțiuni`

**Exemplul 9.4** – **vezi curs**

## 9.5. Excepții externe

- Excepțiile externe:
  - sunt excepții definite de utilizator;
  - sunt declarate în secțiunea declarativă a unui bloc, subprogram sau pachet;
  - sunt activate explicit în partea executabilă a blocului (folosind comanda *RAISE* însoțită de numele excepției);
  - pot să apară în toate secțiunile unui bloc, subprogram sau pachet;
  - nu pot să apară în instrucțiuni de atribuire sau în comenzi *SQL*;
  - în mod implicit, au asociat același cod (+1) și același mesaj (*User-Defined Exception*).
- Sintaxa de declarare și prelucrare a excepțiilor externe :

```
DECLARE
    nume_excepție EXCEPTION; -- declarare excepție
BEGIN
    ...
    RAISE nume_excepție; -- activare excepție
    -- execuția este întreruptă și se transferă
    -- controlul în secțiunea EXCEPTION
```



```
...  
EXCEPTION  
    WHEN nume_excepție THEN  
        -- definire mod de tratare a erorii  
    ...  
END;
```

**Exemplul 9.5 – vezi curs**

- Activarea unei excepții externe poate fi făcută și cu ajutorul procedurii *RAISE\_APPLICATION\_ERROR*, furnizată de pachetul *DBMS\_STANDARD*.
  - Această procedură poate fi utilizată pentru a întoarce un mesaj de eroare unității care o apelează, mesaj mai descriptiv (non standard) decât identificatorul erorii.
  - Are următoarea specificație:

```
RAISE_APPLICATION_ERROR (num NUMBER,  
    msg VARCHAR2, keeperrorstack BOOLEAN);
```

- parametrul *num* reprezintă codul asociat erorii, un număr cuprins între –20000 și –20999;
- parametrul *msg* reprezintă mesajul asociat erorii, un șir de caractere de maxim 2048 bytes;
- parametrul *keeperrorstack* este opțional; dacă are valoarea *TRUE*, atunci noua eroare se va adăuga listei erorilor existente, iar dacă este *FALSE* (valoare implicită) atunci noua eroare va înlocui lista curentă a erorilor (se reține ultimul mesaj de eroare).



- ❖ O aplicație poate apela procedura *RAISE\_APPLICATION\_ERROR* numai dintr-un subprogram stocat (sau metodă).
  - ❖ Dacă procedura *RAISE\_APPLICATION\_ERROR* este apelată, atunci subprogramul se termină și sunt întoarse codul și mesajul asociate erorii respective.
- Procedura *RAISE\_APPLICATION\_ERROR* poate fi apelată în secțiunea executabilă, în secțiunea de tratare a excepțiilor sau simultan în ambele secțiuni.
    - În secțiunea executabilă:

```
DELETE FROM produse WHERE denumire = 'produs';  
IF SQL%NOTFOUND THEN  
    RAISE_APPLICATION_ERROR(-20001, 'Date incorecte');  
END IF;
```

- În secțiunea de tratare a excepțiilor:

```
EXCEPTION
  WHEN exceptie THEN
    RAISE_APPLICATION_ERROR(-20002,'info invalida');
END;
```

- În ambele secțiuni:

```
DECLARE
  exceptie EXCEPTION;
  PRAGMA EXCEPTION_INIT (exceptie, -20000);
BEGIN
  DELETE FROM produse WHERE denumire = 'produs cautat';
  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20000,'Date incorecte');
  END IF;
EXCEPTION
  WHEN exceptie THEN
    DBMS_OUTPUT.PUT_LINE('Mesajul exceptiei: '||SQLERRM);
    DBMS_OUTPUT.PUT_LINE('Codul exceptiei: '||SQLCODE);
END;
```



- ❖ Procedura *RAISE\_APPLICATION\_ERROR* facilitează comunicarea dintre *client* și *server*, transmițând aplicației *client* erori specifice aplicației de pe *server* (de obicei, un *trigger*).
- ❖ Procedura *RAISE\_APPLICATION\_ERROR* este doar un mecanism folosit pentru comunicarea dintre *server* și *client* a unei erori definite de utilizator, care permite ca procesul *client* să trateze excepția.

#### Exemplul 9.6

```
CREATE OR REPLACE TRIGGER trig
  BEFORE UPDATE OF serie ON case
  FOR EACH ROW
  WHEN (NEW.serie <> OLD.serie)
BEGIN
  RAISE_APPLICATION_ERROR (-20145,
    'Nu puteti modifica seria casei fiscale!');
END;
/

--blocul urmator detecteaza si trateaza eroarea
DECLARE
  -- declarare exceptie
  exceptie EXCEPTION;
  -- asociere un nume codului de eroare folosit in trigger
  PRAGMA EXCEPTION_INIT(exceptie,-20145);
```

```
BEGIN
  -- lansare comanda declasatoare
  UPDATE case
  SET  serie = serie||'_';
EXCEPTION
  -- tratare exceptie
  WHEN exceptie THEN
    -- se afiseaza mesajul erorii specificat in trigger
    -- in procedura RAISE_APPLICATION_ERROR
    DBMS_OUTPUT.PUT_LINE (SQLERRM);
END;
/
```

## 9.6. Cazuri speciale în tratarea excepțiilor

- Dacă se declanșează o excepție într-un bloc simplu atunci:
  - execuția blocului este întreruptă (setul de comenzi care urmează după comanda care a declanșat excepția nu se mai execută);
  - controlul este transferat în secțiunea de tratare a excepțiilor;
  - se tratează excepția (se execută comenzile specificate în *handler*-ul excepției respective);
  - se iese din bloc.



- ❖ Dacă după o eroare se dorește totuși continuarea prelucrării datelor, este suficient ca instrucțiunea care a declanșat excepția să fie inclusă într-un subbloc.
- ❖ În acest caz, după tratarea excepției și ieșirea din subbloc, se continuă secvența de instrucțiuni din blocul principal.

```
BEGIN
  comanda_1;  -- declanseaza exceptia E
  comanda_2;  -- nu se mai executa
EXCEPTION
  WHEN E THEN
    set_comenzi; -- se executa
END;
```

- Dacă se dorește execuția comenzii *comanda\_2* chiar și atunci când *comanda\_1* declanșează o excepție, atunci *comanda\_1* se include într-un subbloc, iar excepția declanșată de aceasta este tratată în acel subbloc.

```

BEGIN
    BEGIN
        comanda_1;  -- declanseaza exceptia E
    EXCEPTION
        WHEN E THEN
            set_comenzi; -- se executa
    END;

    comanda_2;  -- se executa
    EXCEPTION
        ...
END;

```

- Uneori este dificil de aflat care comandă *SQL* a determinat o anumită eroare, deoarece există o singură secțiune pentru tratarea erorilor unui bloc.

Variante de rezolvare a acestor situații:

- utilizarea unui contor care să identifice instrucțiunea *SQL* care a declanșat excepția.

```

DECLARE
    contor NUMBER(1);
BEGIN
    contor :=1;
    comanda_SELECT_1;
    contor :=2;
    comanda_SELECT_2;
    contor :=3;
    comanda_SELECT_3;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('comanda SELECT ' || contor);
END;

```

- Introducerea fiecărei instrucțiuni *SQL* într-un subbloc

```

BEGIN
    BEGIN
        comanda_SELECT_1;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('comanda SELECT 1');
    END;

    BEGIN
        comanda_SELECT_2;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('comanda SELECT 2');
    END;

```

```
BEGIN
    comanda_SELECT_3;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('comanda SELECT 3');
END;
END;
```

## 9.7. Activarea excepțiilor

- Există două metode de activare a unei excepții:
  - activarea explicită a excepției (definită de utilizator sau predefinită) în interiorul blocului, cu ajutorul comenzii *RAISE*;
  - activarea automată a excepției asociate unei erori *Oracle*.
- Excepțiile pot fi generate în oricare dintre secțiunile unui bloc *PL/SQL* (în secțiunea declarativă, în secțiunea executabilă sau în secțiunea de tratare a excepțiilor).
  - La aceste niveluri ale programului, o excepție poate fi gestionată în moduri diferite.
- Pentru a reinvoca o excepție, după ce a fost tratată în blocul curent, se folosește instrucțiunea *RAISE*, dar fără a fi însoțită de numele excepției.
  - În acest fel, după executarea instrucțiunilor corespunzătoare tratării excepției, aceasta se transmite și blocului „părinte“.
  - Pentru a fi recunoscută ca atare de către blocul „părinte“, excepția trebuie să nu fie definită în blocul curent, ci în blocul „părinte“ (sau chiar mai sus în ierarhie), în caz contrar ea putând fi captată de către blocul „părinte“ doar la categoria *OTHERS*.
- Pentru a executa același set de acțiuni în cazul mai multor excepții nominalizate explicit, în secțiunea de tratare a excepțiilor se poate utiliza operatorul *OR* (*WHEN excepție\_1 OR excepție\_2 THEN ...*).
- Pentru a evita tratarea fiecărei erori în parte, se folosește secțiunea *WHEN OTHERS* care va cuprinde acțiuni pentru fiecare excepție care nu a fost tratată, adică pentru captarea excepțiilor neprevăzute sau necunoscute.
  - Această secțiune trebuie utilizată cu atenție deoarece poate masca erori critice sau poate împiedica aplicația să răspundă în mod corespunzător.

## 9.8. Propagarea excepțiilor

- Dacă este declanșată o eroare în secțiunea executabilă, iar
  - blocul curent are un *handler* pentru tratarea acesteia, atunci blocul curent se termină cu succes și controlul este transmis blocului imediat exterior;
  - blocul curent nu are un *handler* pentru tratarea acesteia, atunci excepția se propagă spre blocul „părinte“, iar blocul curent se termină fără succes;
    - procesul se repetă până când fie se găsește într-un bloc modalitatea de tratare a erorii, fie se oprește execuția și se semnalează situația apărută (*unhandled exception error*).
- Dacă este declanșată o eroare în secțiunea declarativă a blocului sau în secțiunea de tratare a erorilor, atunci aceasta este propagată către blocul imediat exterior, chiar dacă există un *handler* al acesteia în blocul curent.
- La un moment dat, într-o secțiune *EXCEPTION*, poate fi activă numai o singură excepție.



❖ 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.

### Exemplul 9.7

```
DECLARE
  v_den  produse.denumire%TYPE;
  v_id   produse.id_produs%TYPE := &p_id;
BEGIN
  SELECT denumire
  INTO   v_den
  FROM   produse
  WHERE  id_produs = v_id;
  <<eticheta>>
  DBMS_OUTPUT.PUT_LINE('Denumirea produsului este '||v_den);
EXCEPTION
  WHEN NO_DATA_FOUND THEN v_den := ' ';
    GOTO eticheta; --salt ilegal in blocul curent
END;
/
```

### 9.8.1 Excepție declanșată în secțiunea executabilă

- Excepția este produsă și tratată în subbloc. După tratarea excepției controlul este transmis blocului exterior.

```
DECLARE
  A EXCEPTION;
BEGIN
  ...
  BEGIN
    RAISE A; -- in subbloc se produse exceptia A
  EXCEPTION
    WHEN A THEN ...-- exceptia A este tratata in subbloc
  ...
  END;
-- aici este reluat controlul
END;
```

- Excepția este produsă în subbloc, dar nu este tratată în acesta. Excepția se propagă spre blocul exterior. Regula poate fi aplicată de mai multe ori.

```
DECLARE
  A EXCEPTION;
  B EXCEPTION;
BEGIN
  BEGIN
    RAISE B; -- in subbloc se produse exceptia B
  EXCEPTION
    WHEN A THEN ...
    --exceptia B nu este tratata in subbloc
  END;
EXCEPTION
  WHEN B THEN ...
  /*exceptia B s-a propagat spre blocul exterior unde este
    tratata, apoi controlul este dat in exteriorul blocului */
END;
```

### 9.8.2 Excepție declanșată în secțiunea declarativă

- Dacă în secțiunea declarativă este generată o excepție, atunci aceasta se propagă către blocul exterior, unde are loc tratarea acesteia. Chiar dacă există un *handler* pentru excepție în blocul curent, acesta nu este executat.

**Exemplul 9.8**

```
BEGIN
  DECLARE
    nr_produce    NUMBER(10) := ' ';
    -- este generata eroarea VALUE_ERROR
  BEGIN
    SELECT  COUNT (DISTINCT id_produc)
    INTO    nr_produce
    FROM    facturi_produce;
  EXCEPTION
    WHEN VALUE_ERROR THEN
      -- eroarea nu este captata si tratata in blocul intern
      DBMS_OUTPUT.PUT_LINE('Eroare bloc intern: ' || SQLERRM);
  END;
EXCEPTION
  WHEN VALUE_ERROR THEN
    -- eroarea este captata si tratata in blocul extern
    DBMS_OUTPUT.PUT_LINE('Eroare bloc extern: ' || SQLERRM );
END;
/
```

**9.8.3 Excepție declanșată în secțiunea *EXCEPTION***

- Dacă excepția este declanșată în secțiunea *EXCEPTION*, atunci aceasta se propagă imediat spre blocul exterior.

```
BEGIN
  DECLARE
    A  EXCEPTION;
    B  EXCEPTION;
  BEGIN
    RAISE A; -- este generata exceptia A
  EXCEPTION
    WHEN A THEN
      RAISE B; -- este generata exceptia B
    WHEN B THEN ...
    /* exceptia este propagata spre blocul exterior
       cu toate ca exista aici un handler pentru ea */
  END;
EXCEPTION
  WHEN B THEN ...
  --exceptia B este tratata in blocul exterior
END;
```



## 9.9. Informații despre erori

- Pentru a obține textul corespunzător erorilor apărute la **compilare**, poate fi utilizată vizualizarea *USER\_ERRORS* din dicționarul datelor.
- Pentru informații adiționale referitoare la erori pot fi consultate vizualizările *ALL\_ERRORS* sau *DBA\_ERRORS*.
- Vizualizarea *USER\_ERRORS* oferă informații despre obiectele care au generat erori la compilare; de exemplu:
  - numele obiectului (*NAME*);
  - tipul obiectului (*TYPE*);
  - numărul liniei din codul sursă la care a apărut eroarea (*LINE*);
  - poziția din linie (*POSITION*);
  - mesajul asociat erorii (*TEXT*).



- ❖ *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*).

### Exemplul 9.9

```
CREATE OR REPLACE FUNCTION f_test
RETURN NUMBER;
IS
BEGIN
    RETURN 1;
END;
/
FUNCTION F_TEST compiled
Errors: check compiler log
```

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

LINE	POSITION	TEXT
----	-----	-----
3	1	PLS-00103: Encountered the symbol "IS"

## Bibliografie

1. Connolly T.M., Begg C.E., *Database Systems: A Practical Approach to Design, Implementation and Management*, 5th edition, Pearson Education, 2005
2. Dollinger R., Andron L., *Baze de date și gestiunea tranzacțiilor*, Editura Albastră, Cluj-Napoca, 2004
3. Oracle and/or its affiliates, *Oracle Database Concepts*, 1993, 2021
4. Oracle and/or its affiliates, *Oracle Database Performance Tuning Guide*, 2013, 2023
5. Oracle and/or its affiliates, *Oracle Database SQL Language Reference*, 1996, 2023
6. Oracle and/or its affiliates, *Oracle Database PL/SQL Language Reference*, 1996, 2023
7. Oracle and/or its affiliates, *Oracle Database Administrator's Guide*, 2001, 2023
8. Oracle and/or its affiliates, *Pro\*C/C++ Programmer's Guide*, 1996, 2014
9. Oracle University, *Oracle Database 11g: PL/SQL Fundamentals, Student Guide*, 2009
10. Popescu I., Alecu A., Velcescu L., Florea (Mihai) G., *Programare avansată în Oracle9i*, Ed. Tehnică, 2004