10. Tratarea erorilor în PL/SQL

Mecanismul de gestiune a erorilor permite utilizatorului să definească şi să controleze comportamentul programului atunci când acesta generează o eroare. În acest fel, aplicaţia nu este oprită, revenind într-un regim normal de execuţie.

Într-un program *PL/SQL* pot să apară erori la compilare sau erori la execuţie. Erorile care apar în timpul compilării sunt detectate de motorul *PL/SQL* şi sunt comunicate programatorului care va face corecţia acestora. Programul nu poate trata aceste erori deoarece nu a fost încă executat. Erorile care apar în timpul execuţiei nu mai sunt tratate interactiv. În program trebuie prevăzută apariţia unei astfel de erori şi specificat modul concret de tratare a acesteia. Atunci când apare eroarea este declanşată o excepţie, iar controlul trece la o secţiune separată a programului, unde va avea loc tratarea erorii.

Gestiunea erorilor în *PL/SQL* face referire la conceptul de excepţie. Excepţia este un eveniment particular (eroare sau avertisment) generat de *server*-ul *Oracle* sau de aplicaţie, care necesită o tratare specială. În *PL/SQL* mecanismul de tratare a excepţiilor permite programului să îşi continue execuţia şi în prezenţa anumitor erori.

Excepţiile pot fi definite, activate, tratate la nivelul fiecărui bloc din program (program principal, funcţii şi proceduri, blocuri interioare acestora). Execuţia unui bloc se termină întotdeauna atunci când apare o excepţie, dar se pot executa acţiuni ulterioare apariţiei acesteia, într-o secţiune specială de tratare a excepţiilor.

Posibilitatea de a da nume fiecărei excepţii, de a izola tratarea erorilor într-o secţiune particulară, de a declanşa automat erori (în cazul excepţiilor interne) îmbunătăţeşte lizibilitatea şi fiabilitatea programului. Prin utilizarea excepţiilor şi rutinelor de tratare a excepţiilor, un program *PL/SQL* devine robust şi capabil să trateze atât erorile aşteptate, cât şi cele neaşteptate ce pot apărea în timpul execuţiei.

Secţiunea de tratare a erorilor

Pentru a gestiona excepţiile, utilizatorul trebuie să scrie câteva comenzi care preiau controlul derulării blocului *PL/SQL*. Aceste comenzi sunt situate în secţiunea de tratare a erorilor dintr-un bloc *PL/SQL* şi sunt cuprinse între cuvintele cheie *EXCEPTION* şi *END*, conform următoarei sintaxe generale:

EXCEPTION
 WHEN *nume_excepţie1* [***OR*** *nume_excepţie2 ...*] ***THEN***
  secvenţa_de_instrucţiuni_1;
 [***WHEN*** *nume_excepţie3* [***OR*** *nume_excepţie4 ...*] ***THEN***
  secvenţa_de_instrucţiuni_2;]
 ...

[***WHEN OTHERS THEN***
secvenţa_de_instrucţiuni_n;]
END*;*

De remarcat că *WHEN OTHERS* trebuie să fie ultima clauză şi trebuie să fie unică. Toate excepţiile care nu au fost analizate vor fi tratate prin această clauză. Evident, în practică nu se utilizează forma *WHEN OTHERS THEN NULL*.

În *PL/SQL* există două tipuri de excepţii:

- excepţii interne, care se produc atunci când un bloc *PL/SQL* nu respectă o regulă *Oracle* sau depăşeşte o limită a sistemului de operare;
- excepţii externe definite de utilizator (*user-defined error*), care sunt declarate în secţiunea declarativă a unui bloc, subprogram sau pachet şi care sunt activate explicit în partea executabilă a blocului *PL/SQL*.

Excepţiile interne *PL/SQL* sunt de două tipuri:

- excepţii interne predefinite (*predefined Oracle Server error*);
- excepţii interne nepredefinite (*non-predefined Oracle Server error*).

Funcţii pentru identificarea excepţiilor

Indiferent de tipul excepţiei, aceasta are asociate două elemente:

- un cod care o identifică;
- un mesaj cu ajutorul căruia se poate interpreta excepţia respectivă.

Cu ajutorul funcţiilor *SQLCODE* şi *SQLERRM* se pot obţine codul şi mesajul asociate excepţiei declanşate. Lungimea maximă a mesajului este de 512 caractere.

De exemplu, pentru eroarea predefinită *ZERO_DIVIDE*, codul *SQLCODE* asociat este -1476, iar mesajul corespunzător erorii, furnizat de *SQLERRM*, este „*divide by zero error*“.

Codul erorii este:

- un număr negativ, în cazul unei erori sistem;
- numărul +100, în cazul excepţiei *NO_DATA_FOUND*;
- numărul 0, în cazul unei execuţii normale (fără excepţii);
- numărul 1, în cazul unei excepţii definite de utilizator.

Funcţiile *SQLCODE* şi *SQLERRM* nu se pot utiliza direct ca parte a unei instrucţiuni *SQL*. Valorile acestora trebuie atribuite unor variabile locale. Rezultatul funcţiei *SQLCODE* poate fi asignat unei variabile de tip numeric, iar cel al funcţiei *SQLERRM* unei variabile de tip caracter. Variabilele locale astfel definite pot fi utilizate în comenzi *SQL*.

***Exemplu*:**

Să se scrie un bloc *PL/SQL* prin care să se exemplifice situaţia comentată.

```
DECLARE
  eroare_cod    NUMBER;
  eroare_mesaj  VARCHAR2(100);
BEGIN
  …
EXCEPTION
  …
  WHEN OTHERS THEN
    eroare_cod := SQLCODE;
    eroare_mesaj := SUBSTR(SQLERRM,1,100);
    INSERT  INTO erori
    VALUES  (eroare_cod, eroare_mesaj);
END;
```

Mesajul asociat excepţiei declanşate poate fi furnizat şi de funcţia *DBMS_UTILITY.FORMAT_ERROR_STACK*.

Excepţii interne

Excepţiile interne se produc atunci când un bloc *PL/SQL* nu respectă o regulă *Oracle* sau depăşeşte o limită a sistemului de exploatare.

Aceste excepţii 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*, excepţia asociată ei se declanşează implicit. De exemplu, dacă apare eroarea *ORA-01403* (deoarece o comandă *SELECT* nu returnează nici o linie), atunci implicit *PL/SQL* activează excepţia *NO_DATA_FOUND*. Cu toate că fiecare astfel de excepţie are asociat un cod specific, ele trebuie referite prin nume.

Excepţii interne predefinite

Excepţiile interne predefinite nu trebuie declarate în secţiunea declarativă şi sunt tratate implicit de către *server*-ul *Oracle*. Ele sunt referite prin nume (*CURSOR_ALREADY_OPEN*, *DUP_VAL_ON_INDEX*, *NO_DATA_FOUND* etc.). *PL/SQL* declară aceste excepţii în pachetul *DBMS_STANDARD*.

Excepţii predefinite

Nume excepţie	**Cod eroare *Oracle***	**Descriere**
ACCES_INTO_NULL	*ORA-06530*	Asignare de valori atributelor unui obiect neiniţializat.
CASE_NOT_FOUND	*ORA-06592*	Nu este selectată nici una din clauzele *WHEN* ale lui *CASE* şi nu există nici clauza *ELSE* (excepţie specifică lui *Oracle9i*).
COLLECTION_IS_NULL	*ORA-06531*	Aplicarea unei metode (diferite de *EXISTS*) unui tabel imbricat sau unui vector neiniţializat.
CURSOR_ALREADY_OPEN	*ORA-06511*	Deschiderea unui cursor care este deja dechis.
DUP_VAL_ON_INDEX	*ORA-00001*	Detectarea unei dubluri într-o coloană unde acestea sunt interzise.
INVALID_CURSOR	*ORA-01001*	Operaţie ilegală asupra unui cursor.
INVALID_NUMBER	*ORA-01722*	Conversie nepermisă de la tipul şir de caractere la număr.
LOGIN_DENIED	*ORA-01017*	Nume sau parolă incorecte.
NO_DATA_FOUND	*ORA-01403*	Comanda *SELECT* nu returnează nici o înregistrare.
NOT_LOGGED_ON	*ORA-01012*	Programul *PL/SQL* apelează baza fără să fie conectat la *Oracle*.
SELF_IS_NULL	*ORA-30625*	Apelul unei metode când instanţa este *NULL*.
PROGRAM_ERROR	*ORA-06501*	*PL/SQL* are o problemă internă.
ROWTYPE_MISMATCH	*ORA-*	Incompatibilitate între

	06504	parametrii actuali şi formali, la deschiderea unui cursor parametrizat.
STORAGE_ERROR	*ORA-06500*	*PL/SQL* are probleme cu spaţiul de memorie.
SUBSCRIPT_BEYOND_COUNT	*ORA-06533*	Referire la o componentă a unui *nested table* sau *varray*, folosind un index mai mare decât numărul elementelor colecţiei respective.
SUBSCRIPT_OUTSIDE_LIMIT	*ORA-06532*	Referire la o componentă a unui tabel imbricat sau vector, folosind un index care este în afara domeniului (de exemplu, -1).
SYS_INVALID_ROWID	*ORA-01410*	Conversia unui şir de caractere într-un *ROWID* nu se poate face deoarece şirul nu reprezintă un *ROWID* valid.
TIMEOUT_ON_RESOURCE	*ORA-00051*	Expirarea timpului de aşteptare pentru eliberarea unei resurse.
TRANSACTION_BACKED_OUT	*ORA-00061*	Tranzacţia a fost anulată datorită unei interblocări.
TOO_MANY_ROWS	*ORA-01422*	*SELECT…INTO* întoarce mai multe linii.
VALUE_ERROR	*ORA-06502*	Apariţia unor erori în conversii, constrângeri sau erori aritmetice.
ZERO_DIVIDE	*ORA-01476*	Sesizarea unei împărţiri la zero.

***Exemplu*:**

6

Să se scrie un bloc *PL/SQL* prin care să se afişeze numele artiştilor de o anumită naţionalitate care au fotografii expuse în săli.

1) Dacă rezultatul interogării returnează mai mult decât o linie, atunci să se trateze excepţia şi să se insereze în tabelul *mesaje* textul „mai mulţi creatori“.
2) Dacă rezultatul interogării nu returnează nici o linie, atunci să se trateze excepţia şi să se insereze în tabelul *mesaje* textul „nici un creator“.
3) Dacă rezultatul interogării este o singură linie, atunci să se insereze în tabelul *mesaje* numele artistului şi pseudonimul acestuia.
4) Să se trateze orice altă eroare, inserând în tabelul *mesaje* textul „alte erori au apărut“.

```
SET VERIFY OFF
ACCEPT national PROMPT 'Introduceti nationalitatea:'
DECLARE
  v_nume_artist   artist.nume%TYPE;
  v_pseudonim     artist.pseudonim%TYPE;
  v_national
artist.nationalitate%TYPE:='&national';
BEGIN
  SELECT   nume, pseudonim
  INTO     v_nume_artist, v_pseudonim
  FROM     artist
  WHERE    nationalitate = v_national;
  INSERT   INTO mesaje (rezultat)
  VALUES   (v_nume_artist||'-'||v_pseudonim);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
INSERT   INTO mesaje (rezultat)
VALUES   ('nici un creator');
  WHEN TOO_MANY_ROWS THEN
INSERT   INTO mesaje (rezultat)
VALUES   ('mai multi creatori');
  WHEN OTHERS THEN
INSERT   INTO mesaje (rezultat)
VALUES   ('alte erori au aparut');
END;
/
SET VERIFY ON
```

Aceeaşi excepţie poate să apară în diferite circumstanţe. De exemplu,

excepţia *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 situaţii, este greu de stabilit care dintre ele a generat eroarea şi este necesară restructurarea blocului, astfel încât acesta să poată diferenţia cele două situaţii.

Excepţii interne nepredefinite

Excepţiile interne nepredefinite sunt declarate în secţiunea declarativă şi sunt tratate implicit de către *server*-ul *Oracle*. Ele pot fi gestionate prin clauza *OTHERS*, în secţiunea *EXCEPTION*.

Diferenţierea acestor erori este posibilă doar cu ajutorul codului. După cum s-a mai specificat, codul unei excepţii interne este un număr negativ, în afară de excepţia *NO_DATA_FOUND*, care are codul +100.

O altă metodă pentru tratarea unei erori interne nepredefinite (diferită de folosirea clauzei *OTHERS* drept detector universal de excepţii) este utilizarea directivei de compilare (pseudo-instrucţiune) *PRAGMA EXCEPTION_INIT*. Această directivă permite asocierea numelui unei excepţii cu un cod de eroare intern. În felul acesta, orice excepţie internă poate fi referită printr-un nume şi se pot scrie rutine speciale pentru tratarea acesteia. Directiva este procesată în momentul compilării, şi nu la execuţie. Directiva trebuie să apară în partea declarativă a unui bloc, pachet sau subprogram, după definirea numelui excepţiei. *PRAGMA EXCEPTION_INIT* poate să apară de mai multe ori într-un program. De asemenea, pot fi asignate mai multe nume pentru acelaşi cod de eroare.

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

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

nume_excepţie ***EXCEPTION****;*

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

PRAGMA EXCEPTION_INIT (*nume_excepţie*, *cod_eroare*)*;*

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

***Exemplu*:**

Dacă există fotografii create de un anumit artist, să se tipărească un mesaj prin care utilizatorul este anunţat că artistul respectiv nu poate fi şters din baza de date (violarea constrângerii de integritate având codul eroare *Oracle* -2292).

8

```
SET VERIFY OFF
DEFINE p_nume = Ionescu
DECLARE
  fotografia_exista  EXCEPTION;
  PRAGMA EXCEPTION_INIT(fotografia_exista,-2292);
BEGIN
  DELETE FROM artist WHERE nume = '&p_nume';
  COMMIT;
EXCEPTION
  WHEN fotografia_exista THEN
    DBMS_OUTPUT.PUT_LINE ('nu puteti sterge artistul
cu numele ' || '&p_nume' || ' deoarece exista
fotografii create de acesta');
END;
/
SET VERIFY ON
```

Excepţii externe

PL/SQL permite utilizatorului să definească propriile sale excepţii. Aceste excepţii pot să apară în toate secţiunile unui bloc, subprogram sau pachet. Excepţiile externe sunt definite în partea declarativă a blocului, deci posibilitatea de referire la ele este asigurată. În mod implicit, toate excepţiile externe au asociat acelaşi cod (+1) şi acelaşi mesaj (*USER DEFINED EXCEPTION*).

Tratarea unei astfel de erori se face într-o manieră similară modului de tratare descris anterior. Activarea excepţiei externe este făcută explicit, folosind comanda *RAISE* însoţită de numele excepţiei. Comanda opreşte execuţia normală a blocului *PL/SQL* şi transferă controlul „administratorului“ excepţiilor.

Declararea şi prelucrarea excepţiilor externe respectă următoarea sintaxă:

DECLARE
nume_excepţie ***EXCEPTION***; -- *declarare excepţie*
BEGIN
...
RAISE *nume_excepţie*; *--declanşare excepţie*
-- codul care urmează nu mai este executat
...
EXCEPTION
WHEN *nume_excepţie* ***THEN***
-- definire mod de tratare a erorii

...

END;

Excepţiile trebuie privite ca nişte variabile, în sensul că ele sunt active în secţiunea în care sunt declarate. Ele nu pot să apară în instrucţiuni de atribuire sau în comenzi *SQL*.

Este recomandat ca fiecare subprogram să aibă definită o zonă de tratare a excepţiilor. Dacă pe parcursul execuţiei programului intervine o eroare, atunci acesta generează o excepţie şi controlul se transferă blocului de tratare a erorilor.

***Exemplu*:**

Să se scrie un bloc *PL/SQL* care afişează numărul creatorilor fotografiilor care au valoarea mai mare sau mai mică cu 100000$ decât o valoare specificată. Să se tipărească un mesaj adecvat, dacă nu există nici un artist care îndeplineşte această condiţie.

```
VARIABLE g_mesaj VARCHAR2(100)
SET VERIFY OFF
ACCEPT p_val PROMPT 'va rog specificati valoarea:'
DECLARE
  v_val       fotografie.valoare%TYPE := &p_val;
  v_inf       fotografie.valoare%TYPE := v_val -
100000;
  v_sup       fotografie.valoare%TYPE := v_val +
100000;
  v_numar     NUMBER(7);
  e_nimeni    EXCEPTION;
  e_mai_mult  EXCEPTION;
BEGIN
  SELECT  COUNT(DISTINCT cod_artist)
  INTO    v_numar
  FROM    fotografie
  WHERE   valoare 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 artist cu valoarea
```

```
            fotografiilor cuprinsa intre '||v_inf ||'
si '||v_sup;
  WHEN e_mai_mult THEN
    :g_mesaj:='exista '||v_numar||' artisti cu
valoarea fotografiilor cuprinsa intre '||v_inf||' si
'||v_sup;
  WHEN OTHERS THEN
    :g_mesaj:='au aparut alte erori';
END;
/

SET VERIFY ON
PRINT g_mesaj
```

Activarea unei excepţii externe poate fi făcută şi cu ajutorul procedurii *RAISE_APPLICATION_ERROR*, furnizată de pachetul *DBMS_STANDARD*.

RAISE_APPLICATION_ERROR poate fi folosită pentru a returna un mesaj de eroare unităţii care o apelează, mesaj mai descriptiv decât identificatorul erorii. Unitatea apelantă poate fi *SQL*Plus*, un subprogram *PL/SQL* sau o aplicaţie *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 şi –20999, specificat de utilizator pentru excepţia respectivă, iar *mesaj_eroare* este un text asociat erorii, care poate avea maximum 2048 octeţi.

Parametrul boolean este opţional. Dacă acest parametru este *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.

O aplicaţie poate apela *RAISE_APPLICATION_ERROR* numai dintr-un subprogram stocat (sau metodă). Dacă procedura *RAISE_APPLICATION_ERROR* este apelată, atunci subprogramul se termină şi sunt returnate codul şi mesajul asociate erorii respective.

Procedura *RAISE_APPLICATION_ERROR* poate fi folosită în secţiunea executabilă, în secţiunea de tratare a erorilor şi chiar simultan în ambele secţiuni.

- În secţiunea executabilă:

```
DELETE FROM fotografie WHERE material = 'hartie';
IF SQL%NOTFOUND THEN
  RAISE_APPLICATION_ERROR(-20201,'info incorecta');
```

```
END IF;
```

- În secţiunea de tratare a erorilor:

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20202,'info invalida');
END;
```

- În ambele secţiuni:

```
DECLARE
  e_material  EXCEPTION;
  PRAGMA EXCEPTION_INIT (e_material, -20777);
BEGIN
  …
  DELETE FROM fotografie WHERE valoare < 100001;
  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20777,
       'nu exista fotografie cu aceasta valoare');
  END IF;
EXCEPTION
  WHEN e_material THEN
    -- trateaza eroarea aceasta
  …
END;
```

RAISE_APPLICATION_ERROR facilitează comunicaţia dintre *client* şi *server*, transmiţând aplicaţiei *client* erori specifice aplicaţiei de pe *server* (de obicei, un declanşator). Prin urmare, procedura este doar un mecanism folosit pentru comunicaţia *server* → *client* a unei erori definite de utilizator, care permite ca procesul *client* să trateze excepţia.

***Exemplu*:**

Să se implementeze un declanşator care nu permite acceptarea în sali a fotografiilor având valoarea mai mică de 100000$.

```
CREATE OR REPLACE TRIGGER minim_valoare
BEFORE INSERT ON fotografie
FOR EACH ROW
BEGIN
  IF :NEW.valoare < 100000 THEN
    RAISE_APPLICATION_ERROR
      (-20005,fotografiile trebuie sa aiba valoare
```

```
                mai mare de 100000$');
  END IF;
END;
```

Pe staţia *client* poate fi scris un program care detectează şi tratează eroarea.

```
DECLARE
  /* declarare excepţie */
  nu_accepta  EXCEPTION;
  /* asociază nume, codului eroare folosit in trigger
*/
  PRAGMA EXCEPTION_INIT(nu_accepta,-20005);
BEGIN
  /* incearca sa inserezi */
  INSERT   INTO fotografie…;
EXCEPTION
  /* tratare exceptie */
  WHEN nu_accepta THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
   /* SQLERRM va returna mesaj din
RAISE_APPLICATION_ERROR */
END;
```

Cazuri speciale în tratarea excepţiilor

Dacă se declanşează o excepţie 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 (instrucţiunea *END*).

Prin urmare, dacă excepţia se propagă spre blocul care include blocul curent, restul acţiunilor executabile din subbloc sunt „pierdute“. Dacă după o eroare se doreşte totuşi continuarea prelucrării datelor, este suficient ca instrucţiunea care a declanşat excepţia să fie inclusă într-un subbloc. După ce subblocul a fost terminat, se continuă secvenţa de instrucţiuni din blocul principal.

***Exemplu*:**

```
BEGIN
  DELETE …
  SELECT …--poate declansa exceptia A
           --nu poate fi efectuat INSERT care urmeaza
  INSERT INTO …
EXCEPTION
```

```
  WHEN A THEN …
END;
```

Deficienţa anterioară se poate rezolva incluzând într-un subbloc comanda *SELECT* care a declanşat excepţia.

```
BEGIN
  DELETE …
   BEGIN
     SELECT …
     …
   EXCEPTION
    WHEN A THEN …
    /* dupa ce se trateaza exceptia A, controlul este
      transferat blocului de nivel superior, de fapt
      comenzii INSERT */
   END;
  INSERT  INTO …
  …
EXCEPTION
…
END;
```

Uneori este dificil de aflat care comandă *SQL* a determinat o anumită eroare, deoarece există o singură secţiune pentru tratarea erorilor unui bloc. Sunt sugerate două soluţii pentru rezolvarea acestei probleme.

1) Introducerea unui contor care să identifice instrucţiunea *SQL*.

```
DECLARE
  v_sel_cont  NUMBER(2):=1;
BEGIN
  SELECT …
  v_sel_cont:=2;
  SELECT …
  v_sel_cont:=3;
  SELECT …
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    INSERT INTO log_table(info)
    VALUES ('comanda SELECT ' || TO_CHAR(v_sel_cont)
|| ' nu gaseste date');
END;
```

2) Introducerea fiecărei instrucţiuni *SQL* într-un subbloc.

```
BEGIN
  BEGIN
    SELECT …
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      INSERT INTO log_table(info)
      VALUES('SELECT 1 nu gaseste date');
  END;
  BEGIN
    SELECT …
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      INSERT INTO log_table(info)
      VALUES('SELECT 2 nu gaseste date');
  END;
  …
END;
```

Activarea excepţiilor

Pentru activarea unei excepţii există două metode:

- activarea explicită a excepţiei (definite de utilizator sau predefinite) în interiorul blocului, cu ajutorul comenzii *RAISE*;
- activarea automată a excepţiei asociate unei erori *Oracle*.

Excepţiile pot fi sesizate în secţiunea executabilă, declarativă sau în cea de tratare a excepţiilor. La aceste niveluri ale programului, o excepţie poate fi gestionată în moduri diferite.

Pentru a reinvoca o excepţie, după ce a fost tratată în blocul curent, se foloseşte instrucţiunea *RAISE*, dar fără a fi însoţită de numele excepţiei. În acest fel, după executarea instrucţiunilor corespunzătoare tratării excepţiei, aceasta se transmite şi blocului „părinte“. Pentru a fi recunoscută ca atare de către blocul „părinte“, excepţia 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 acelaşi set de acţiuni în cazul mai multor excepţii nominalizate explicit, în secţiunea de prelucrare a excepţiilor se poate utiliza operatorul *OR*.

Pentru a evita tratarea fiecărei erori în parte, se foloseşte secţiunea *WHEN*

OTHERS care va cuprinde acţiuni pentru fiecare excepţie care nu a fost tratată, adică pentru captarea excepţiilor neprevăzute sau necunoscute. Această secţiune trebuie utilizată cu atenţie deoarece poate masca erori critice sau poate împiedica aplicaţia să răspundă în mod corespunzător.

Propagarea excepţiilor

Dacă este declanşată o eroare în secţiunea executabilă şi blocul curent are un *handler* pentru tratarea ei, atunci blocul se termină cu succes, iar controlul este dat blocului imediat exterior.

Dacă se produce o excepţie care nu este tratată în blocul curent, atunci excepţia se propagă spre blocul „părinte“, iar blocul *PL/SQL* curent se termină fără succes. Procesul se repetă până când fie se găseşte într-un bloc modalitatea de tratare a erorii, fie se opreşte execuţia şi se semnalează situaţia apărută (*unhandled exception error*).

Dacă este declanşată o eroare în partea declarativă a blocului, aceasta este propagată către blocul imediat exterior, chiar dacă există un *handler* al acesteia în blocul corespunzător secţiunii declarative.

La fel se întâmplă dacă o eroare este declanşată în secţiunea de tratare a erorilor. La un moment dat, într-o secţiune *EXCEPTION*, poate fi activă numai o singură excepţie.

Instrucţiunea *GOTO* nu permite:

- saltul la secţiunea de tratare a unei excepţii;
- saltul de la secţiunea de tratare a unei excepţii, în blocul curent.

Comanda *GOTO* permite totuşi saltul de la secţiunea de tratare a unei excepţii la un bloc care include blocul curent.

***Exemplu*:**

Exemplul următor marchează un salt ilegal în blocul curent.

```
DECLARE
  v_var  NUMBER(10,3);
BEGIN
  SELECT dim2/NVL(valoare,0)
  INTO   v_var
  FROM   fotografie
  WHERE  dim1 > 100;
  <<eticheta>>
  INSERT  INTO politaasig(cod_polita, valoare)
  VALUES  (7531, v_var);
EXCEPTION
  WHEN ZERO_DIVIDE THEN v_var:=0;
```

```
    GOTO <<eticheta>>; --salt ilegal in blocul curent
END;
```

În continuare, vor fi analizate modalităţile de propagare a excepţiilor în cele trei cazuri comentate: excepţii sesizate în secţiunea declarativă, în secţiunea executabilă şi în secţiunea de tratare a erorilor.

Excepţie sesizată în secţiunea executabilă

Excepţia este sesizată şi tratată în subbloc. După aceea, controlul revine blocului exterior.

```
DECLARE
  A  EXCEPTION;
BEGIN
  …
  BEGIN
    RAISE A; -- exceptia A sesizata in subbloc
  EXCEPTION
    WHEN A THEN …-- exceptia tratata in subbloc
    …
  END;
-- aici este reluat controlul
END;
```

Excepţia este sesizată în subbloc, dar nu este tratată în acesta şi atunci se propagă spre blocul exterior. Regula poate fi aplicată de mai multe ori.

```
DECLARE
  A  EXCEPTION;
  B  EXCEPTION;
BEGIN
  BEGIN
    RAISE B; --exceptia B sesizata in subbloc
  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 a fost
   tratata, apoi controlul trece in exteriorul
blocului */
END;
```

Excepţie sesizată în secţiunea declarativă

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

***Exemplu*:**

Să se realizeze un program prin care să se exemplifice propagarea erorilor apărute în secţiunea declarativă a unui bloc *PL/SQL*. Programul calculează numărul artistilor care au fotografii expuse în sali.

```
BEGIN
  DECLARE
    nr_artisti   NUMBER(3) := 'XYZ';
  BEGIN
    SELECT  COUNT (DISTINCT cod_artist)
    INTO    nr_artisti
    FROM    fotografie;
  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;
```

Deoarece la iniţializarea variabilei *nr_artisti* apare o neconcordanţă între tipul declarat şi cel asignat, este generată eroarea internă *VALUE_ERROR*. Cum eroarea a apărut în partea declarativă a blocului intern, deşi acesta conţine un *handler OTHERS* care ar fi putut capta eroarea, *handler*-ul nu este executat, eroarea fiind propagată către blocul extern unde este tratată în *handler*-ul *OTHERS* asociat. Aceasta se poate remarca deoarece la execuţie se obţine mesajul: „*Eroare bloc extern*: *ORA-06502: PL/SQL*: *numeric or value error*“.

Excepţie sesizată în secţiunea *EXCEPTION*

Dacă excepţia este sesizată în secţiunea *EXCEPTION*, ea se propagă imediat spre blocul exterior.

```
BEGIN
  DECLARE
    A  EXCEPTION;
    B  EXCEPTION;
```

```
  BEGIN
    RAISE A; --sesizare exceptie A
  EXCEPTION
    WHEN A THEN
      RAISE B; --sesizare exceptie 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;
```

Informaţii despre erori

Pentru a obţine textul corespunzător erorilor la compilare, poate fi utilizată vizualizarea *USER_ERRORS* din dicţionarul datelor. Pentru informaţii adiţionale referitoare la erori pot fi consultate vizualizările *ALL_ERRORS* sau *DBA_ERRORS*.

Vizualizarea *USER_ERRORS* are câmpurile: *NAME* (numele obiectului), *TYPE* (tipul obiectului), *SEQUENCE* (numărul secvenţei), *LINE* (numărul liniei din codul sursă în care a apărut eroarea), *POSITION* (poziţia în linie unde a apărut eroarea) şi *TEXT* (mesajul asociat erorii).

***Exemplu*:**

Să se afişeze erorile de compilare din procedura *alfa*.

```
SELECT LINE, POSITION, TEXT
FROM   USER_ERRORS
WHERE  NAME = 'ALFA';
```

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