Tratarea erorilor

Tratarea erorilor

Mecanismul de gestiune a erorilor permite utilizatorului sa defineasca si sa controleze comportamentul programului atunci cand acesta genereaza o eroare. In acest fel, aplicatia nu este oprita, revenind intr-un regim normal de executie.

Intr-un program ***PL/SQL*** pot sa apara erori la compilare sau erori la executie.

Erorile care apar in timpul compilarii sunt detectate de motorul ***PL/SQL*** si sunt comunicate programatorului care va face corectia acestora. Programul nu poate trata aceste erori deoarece nu a fost inca executat.

Erorile care apar in timpul executiei nu mai sunt tratate interactiv. In program trebuie prevazuta aparitia unei astfel de erori si specificat modul concret de tratare a acesteia. Atunci cand apare eroarea este declansata o exceptie, iar controlul trece la o sectiune separata a programului, unde va avea loc tratarea erorii.

Gestiunea erorilor in ***PL/SQL*** face referire la conceptul de exceptie. Exceptia este un eveniment particular (eroare sau avertisment) generat de ***server-ul Oracle*** sau de aplicatie, care necesita o tratare speciala. In ***PL/SQL*** mecanismul de tratare a exceptiilor permite programului sa isi continue executia si in prezenta anumitor erori.

Exceptiile pot fi definite, activate, tratate la nivelul fiecarui bloc din program (program principal, functii si proceduri, blocuri interioare acestora). Executia unui bloc se termina intotdeauna atunci cand apare o exceptie, dar se pot executa actiuni ulterioare aparitiei acesteia, intr-o sectiune speciala de tratare a exceptiilor.

Posibilitatea de a da nume fiecarei exceptii, de a izola tratarea erorilor intr-o sectiune particulara, de a declansa automat erori (in cazul exceptiilor interne) imbunatateste lizibilitatea si fiabilitatea programului. Prin utilizarea exceptiilor si rutinelor de tratare a exceptiilor, un program ***PL/SQL*** devine robust si capabil sa trateze atat erorile asteptate, cat si cele neasteptate ce pot aparea in timpul executiei.

Sectiunea de tratare a erorilor

Pentru a gestiona exceptiile, utilizatorul trebuie sa scrie cateva comenzi care preiau controlul derularii blocului ***PL/SQL.*** Aceste comenzi sunt situate in sectiunea de tratare a erorilor dintr-un bloc ***PL/SQL*** si sunt cuprinse intre

cuvintele cheie *EXCEPTION* si *END*, conform urmatoarei sintaxe generale:

EXCEPTION
WHEN*nume_exceptie1* ***[OR*** *nume_exceptie2 ...*] ***THEN*** *secventadeinstructiunil*;
[WHEN*numeexceptie3* ***[OR*** *numeexceptie4 ...*] ***THEN***
secventa_de_instructiuni_2;]

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

De remarcat ca *WHEN OTHERS* trebuie sa fie ultima clauza si trebuie sa fie unica. Toate exceptiile care nu au fost analizate vor fi tratate prin aceasta clauza. Evident, in practica nu se utilizeaza forma *WHEN OTHERS THEN NULL.*

In *PL/SQL* exista doua tipuri de exceptii:

- exceptii interne, care se produc atunci cand un bloc *PL/SQL* nu respecta o regula *Oracle* sau depaseste o limita a sistemului de operare;
- exceptii externe definite de utilizator ***(****user-defined error****),*** care sunt declarate in sectiunea declarativa a unui bloc, subprogram sau pachet si care sunt activate explicit in partea executabila a blocului *PL/SQL****.***

Exceptiile interne *PL/SQL* sunt de doua tipuri:

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

Functii pentru identificarea exceptiilor

Indiferent de tipul exceptiei, aceasta are asociate doua elemente:

- un cod care o identifica;
- un mesaj cu ajutorul caruia se poate interpreta exceptia respectiva.

Cu ajutorul functiilor *SQLCODE* si *SQLERRM* se pot obtine codul si mesajul asociate exceptiei declansate. Lungimea maxima a mesajului este de 512 caractere.

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

Codul erorii este:

- un numar negativ, in cazul unei erori sistem;
- numarul +100, in cazul exceptiei *NODATAFOUND****;***
- numarul 0, in cazul unei executii normale (fara exceptii);

• numarul 1, in cazul unei exceptii definite de utilizator.

Functiile *SQLCODE* si *SQLERRM* nu se pot utiliza direct ca parte a unei instructiuni *SQL***.** Valorile acestora trebuie atribuite unor variabile locale.

Rezultatul functiei *SQLCODE* poate fi asignat unei variabile de tip numeric, iar cel al functiei *SQLERRM* unei variabile de tip caracter. Variabilele locale astfel definite pot fi utilizate in comenzi *SQL***.**

Exemplu:

Sa se scrie un bloc *PL/SQL* prin care sa se exemplifice situatia comentata.

```
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 exceptiei declansate poate fi furnizat si de functia *DBMS_ UTILITY. FORMA TERRORSTACK***.**

Exceptii interne

Exceptiile interne se produc atunci cand un bloc *PL/SQL* nu respecta o regula *Oracle* sau depaseste o limita a sistemului de exploatare.

Aceste exceptii pot fi independente de structura bazei de date sau pot sa apara datorita nerespectarii constrangerilor statice implementate in structura ***(****PRIMARY KEY****,*** *FOREIGN KEY****,*** *NOT NULL, UNIQUE****,*** *CHECK****)***.

Atunci cand apare o eroare *Oracle****,*** exceptia asociata ei se declanseaza implicit. De exemplu, daca apare eroarea *ORA-01403* (deoarece o comanda *SELECT* nu returneaza nici o linie), atunci implicit *PL/SQL* activeaza exceptia *NODATAFOUND.* Cu toate ca fiecare astfel de exceptie are asociat un cod specific, ele trebuie referite prin nume.

Exceptii interne predefinite

Exceptiile interne predefinite nu trebuie declarate in sectiunea declarativa si sunt tratate implicit de catre *server-****ul*** *Oracle***.** Ele sunt referite prin nume

(CURSORALREADY_OPEN, DUP_VAL_ON_INDEX, NO_DATAFOUND etc.). ***PL/SQL*** declara aceste exceptii in pachetul ***DBMSSTANDARD.***

Nume exceptie	Cod eroare	Descriere
ACCESINTONULL	*ORA-06530*	Asignare de valori atributelor unui obiect neinitializat.
CASENOTFO UND	*ORA-06592*	Nu este selectata nici una din clauzele *WHEN* ale lui *CASE* si nu exista nici clauza *ELSE* (exceptie specifica lui *Oracle9i).*
COLLECTIONISNULL	*ORA-06531*	Aplicarea unei metode (diferite de *EXISTS)* unui tabel imbricat sau unui vector neinitializat.
CURSORALREADYOPEN	*ORA-06511*	Deschiderea unui cursor care este deja dechis.
DUPVALONINDEX	*ORA-00001*	Detectarea unei dubluri intr-o coloana unde acestea sunt interzise.
INVALID CURSOR	*ORA-01001*	Operatie ilegala asupra unui cursor.
INVALIDNUMBER	*ORA-01722*	Conversie nepermisa de la tipul sir de caractere la numar.
LOGIN DENIED	*ORA-01017*	Nume sau parola incorecte.
NODATAFOUND	*ORA-01403*	Comanda *SELECT* nu returneaza nici o inregistrare.
NOTLOGGEDON	*ORA-01012*	Programul *PL/SQL* apeleaza baza fara sa fie conectat la *Oracle.*
SELFISNULL	*ORA-30625*	Apelul unei metode cand instanta este *NULL.*
PROGRAM ERROR	*ORA-06501*	*PL/SQL* are o problema interna.
RO WTYPEMISMATCH	*ORA-06504*	Incompatibilitate intre parametrii actuali si formali, la deschiderea unui cursor parametrizat.
STORAGEERROR	*ORA-06500*	*PL/SQL* are probleme cu spatiul de memorie.
SUBSCRIPTBEYONDCO UNT	*ORA-06533*	Referire la o componenta a unui *nested table* sau *varray,* folosind un index mai mare decat numarul elementelor colectiei respective.
SUBSCRIPTO UTSIDELIMIT	*ORA-06532*	Referire la o componenta a unui tabel imbricat sau vector, folosind un index care este in afara domeniului (de exemplu, -1).
SYSINVALIDRO WID	*ORA-01410*	Conversia unui sir de caractere intr-un *ROWID* nu se poate face deoarece sirul nu reprezinta un *ROWID* valid.
TIMEOUTONRESOURCE	*ORA-00051*	Expirarea timpului de asteptare pentru eliberarea unei resurse.
TRANSACTIONBACKEDOUT	*ORA-00061*	Tranzactia a fost anulata datorita unei interblocari.
TOOMANYROWS	*ORA-01422*	*SELECT.INTO* intoarce mai multe linii.

VALUEERROR	*ORA-06502*	Aparitia unor erori in conversii, constrangeri sau erori aritmetice.
ZERO DIVIDE	*ORA-01476*	Sesizarea unei impartiri la zero.

Exemplu:

Sa se scrie un bloc ***PL/SQL*** prin care sa se afiseze numele artistilor de o anumita nationalitate care au opere de arta expuse in muzeu.

1) Daca rezultatul interogarii returneaza mai mult decat o linie, atunci sa se trateze exceptia si sa se insereze in tabelul ***mesaje*** textul „mai multi creatori“.

2) Daca rezultatul interogarii nu returneaza nici o linie, atunci sa se trateze exceptia si sa se insereze in tabelul ***mesaje*** textul „nici un creator“.

3) Daca rezultatul interogarii este o singura linie, atunci sa se insereze in tabelul ***mesaje*** numele artistului si pseudonimul acestuia.

4) Sa se trateze orice alta eroare, inserand in tabelul ***mesaje*** textul „alte erori au aparut“.

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

Aceeasi exceptie poate sa apara in diferite circumstante. De exemplu, exceptia ***NODATAFOUND*** poate fi generata fie pentru ca o interogare nu intoarce un rezultat, fie pentru ca se refera un element al unui tablou ***PL/SQL***

care nu a fost definit (nu are atribuita o valoare). Daca intr-un bloc *PL/SQL* apar ambele situatii, este greu de stabilit care dintre ele a generat eroarea si este necesara restructurarea blocului, astfel incat acesta sa poata diferentia cele doua situatii.

Exceptii interne nepredefinite

Exceptiile interne nepredefinite sunt declarate in sectiunea declarativa si sunt tratate implicit de catre *server-ul Oracle.* Ele pot fi gestionate prin clauza *OTHERS*, in sectiunea *EXCEPTION*.
Diferentierea acestor erori este posibila doar cu ajutorul codului. Dupa cum s-a mai specificat, codul unei exceptii interne este un numar negativ, in afara de exceptia *NODATAFOUND,* care are codul +100.

O alta metoda pentru tratarea unei erori interne nepredefinite (diferita de folosirea clauzei *OTHERS* drept detector universal de exceptii) este utilizarea directivei de compilare (pseudo-instructiune) *PRAGMA EXCEPTIONINIT.* Aceasta directiva permite asocierea numelui unei exceptii cu un cod de eroare intern. In felul acesta, orice exceptie interna poate fi referita printr-un nume si se pot scrie rutine speciale pentru tratarea acesteia. Directiva este procesata in momentul compilarii, si nu la executie.
Directiva trebuie sa apara in partea declarativa a unui bloc, pachet sau subprogram, dupa definirea numelui exceptiei. *PRAGMA EXCEPTION INIT* poate sa apara de mai multe ori intr-un program. De asemenea, pot fi asignate mai multe nume pentru acelasi cod de eroare.

In acest caz, tratarea erorii se face in urmatoarea maniera:

1) se declara numele exceptiei in partea declarativa sub forma:

numeexceptie ***EXCEPTION****;*

2) se asociaza numele exceptiei cu un cod eroare standard *Oracle,* utilizand comanda:

PRAGMA EXCEPTION__INIT *(nume exceptie, coderoare);*

3) se refera exceptia in sectiunea de gestiune a erorilor (exceptia este tratata automat, fara a fi necesara comanda *RAISE).*

Exemplu:
Daca exista opere de arta create de un anumit artist, sa se tipareasca un mesaj prin care utilizatorul este anuntat ca artistul respectiv nu poate fi sters din baza de date (violarea constrangerii de integritate avand codul eroare *Oracle* -2292).

```
SET VERIFY OFF DEFINE p_nume = Monet
```

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

Exceptii externe

PL/SQL permite utilizatorului sa defineasca propriile sale exceptii. Aceste exceptii pot sa apara in toate sectiunile unui bloc, subprogram sau pachet. Exceptiile externe sunt definite in partea declarativa a blocului, deci posibilitatea de referire la ele este asigurata. In mod implicit, toate exceptiile externe au asociat acelasi cod (+1) si acelasi mesaj ***(****USER DEFINED EXCEPTION****)***.

Tratarea unei astfel de erori se face intr-o maniera similara modului de tratare descris anterior. Activarea exceptiei externe este facuta explicit, folosind comanda *RAISE* insotita de numele exceptiei. Comanda opreste executia normala a blocului *PL/SQL* si transfera controlul „admmistratoralui“ exceptiilor.

Declararea si prelucrarea exceptiilor externe respecta urmatoarea sintaxa:

DECLARE
numeexceptie ***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;

Exceptiile trebuie privite ca niste variabile, in sensul ca ele sunt active in sectiunea in care sunt declarate. Ele nu pot sa apara in instructiuni de atribuire sau in comenzi ***SQL.***

Este recomandat ca fiecare subprogram sa aiba definita o zona de tratare a exceptiilor. Daca pe parcursul executiei programului intervine o eroare, atunci acesta genereaza o exceptie si controlul se transfera blocului de tratare a erorilor.

Exemplu:

Sa se scrie un bloc ***PL/SQL*** care afiseaza numarul creatorilor operelor de arta din muzeu care au valoarea mai mare sau mai mica cu 100000$ decat o valoare specificata. Sa se tipareasca un mesaj adecvat, daca nu exista nici un artist care indeplineste aceasta conditie.

```
VARIABLE g_mesaj VARCHAR2(100)
SET VERIFY OFF
ACCEPT p_val PROMPT 'va rog specificati valoarea:'
DECLARE
  v val       opera.valoare%TYPE  := &p_val;
  v_inf       opera.valoare%TYPE  := v val - 100000
  v_sup v     opera.valoare%TYPE  := vval   ;
  numar       NUMBER(7);             +
                                     100000
  e_nimeni e  EXCEPTION;             ;
  mai mult    EXCEPTION;
BEGIN
  SELECT COUNT(DISTINCT cod_autor)
  INTO    v_numar
  FROM    opera
  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
         operelor cuprinsa intre '||v_inf ||' si
  '||v_sup; WHEN e_mai_mult THEN    -   -
    :g_mesaj:='exista '||v_numar||' artisti cu valoarea
         operelor 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 exceptii exteme poate fi facuta si cu ajutorul procedurii *RAISE APPLICATION ERROR*, furnizata de pachetul *DBMS STANDARD.*

RAISEAPPLICATIONERROR poate fi folosita pentru a returna un mesaj de eroare unitatii care o apeleaza, mesaj mai descriptiv (non standard) decat identificatorul erorii. Unitatea apelanta poate fi *SQL*Plus*, un subprogram *PL/SQL* sau o aplicatie *client.*

Procedura are urmatorul antet:

RAISE APPLICATION ERROR *(numareroare* ***IN NUMBER****,*
mesajeroare ***INVARCHAR2****,* [*{****TRUE*** | ***FALSE****}*]);

Atributul *numar eroare* este un numar cuprins intre -20000 si -20999, specificat de utilizator pentru exceptia respectiva, iar *mesaj eroare* este un text asociat erorii, care poate avea maximum 2048 octeti.

Parametrul boolean este optional. Daca acest parametru este *TRUE,* atunci noua eroare se va adauga listei erorilor existente, iar daca este *FALSE* (valoare implicita) atunci noua eroare va inlocui lista curenta a erorilor (se retine ultimul mesaj de eroare).

O aplicatie poate apela *RAISEAPPLICATIONERROR* numai dintr-un subprogram stocat (sau metoda). Daca *RAISE APPLICATION ERROR* este apelata, atunci subprogramul se termina si sunt returnate codul si mesajul asociate erorii respective.

Procedura *RAISE APPLICATION ERROR* poate fi folosita in sectiunea executabila, in sectiunea de tratare a erorilor si chiar simultan in ambele sectiuni.

- In sectiunea executabila:

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

- In sectiunea de tratare a erorilor:

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

- In ambele sectiuni:

```
DECLARE
   e_material EXCEPTION;
   PRAGMA EXCEPTION_INIT (e_material, -20777);
BEGIN
```

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

END;
```

RAISE APPLICATION ERROR faciliteaza comunicatia dintre *client* si *server*, transmitand aplicatiei *client* erori specifice aplicatiei de pe *server* (de obicei, un declansator). Prin urmare, procedura este doar un mecanism folosit pentru comunicatia *server* ^ *client* a unei erori definite de utilizator, care permite ca procesul *client* sa trateze exceptia.

Exemplu:

Sa se implementeze un declansator care nu permite acceptarea in muzeu a operelor de arta avand valoarea mai mica de 100000$.

```
    CREATE OR REPLACE TRIGGER minim_valoare BEFORE INSERT
ON opera FOR EACH ROW BEGIN
    IF :NEW.valoare < 100000 THEN RAISE_APPLICATION_ERROR
    (-20005,'operele de arta trebuie sa aiba valoare mai
mare de 100000$');
    END IF;
    END;
```

Pe statia *client* poate fi scris un program care detecteaza si trateaza eroarea.

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

Cazuri speciale in tratarea exceptiilor

Daca se declanseaza o exceptie intr-un bloc simplu, atunci se face saltul la partea de tratare (***handler***) a acesteia, iar dupa ce este terminata tratarea erorii se iese din bloc (instructiunea ***END).***
Prin urmare, daca exceptia se propaga spre blocul care include blocul curent, restul actiunilor executabile din subbloc sunt „pierdute“. Daca dupa o eroare se doreste totusi continuarea prelucrarii datelor, este suficient ca instructiunea care a declansat exceptia sa fie inclusa intr-un subbloc.
Dupa ce subblocul a fost terminat, se continua secventa de instructiuni 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;
```

Deficienta anterioara se poate rezolva incluzand intr-un subbloc comanda ***SELECT*** care a declansat exceptia.

```
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 comanda ***SQL*** a determinat o anumita eroare, deoarece exista o singura sectiune pentru tratarea erorilor unui bloc. Sunt sugerate doua solutii pentru rezolvarea acestei probleme.

1) Introducerea unui contor care sa identifice instructiunea ***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 fiecarei instructiuni ***SQL*** intr-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 exceptiilor

Pentru activarea unei exceptii exista doua metode:

- activarea explicita a exceptiei (definite de utilizator sau predefinite) in interiorul blocului, cu ajutorul comenzii ***RAISE***;
- activarea automata a exceptiei asociate unei erori ***Oracle.***

Exceptiile pot fi sesizate in sectiunea executabila, declarativa sau in cea de tratare a exceptiilor. La aceste niveluri ale programului, o exceptie poate fi gestionata in moduri diferite.

Pentru a reinvoca o exceptie, dupa ce a fost tratata in blocul curent, se foloseste instructiunea ***RAISE,*** dar fara a fi insotita de numele exceptiei. In acest fel, dupa executarea instructiunilor corespunzatoare tratarii exceptiei, aceasta se

transmite si blocului „parinte“ Pentrn a fi recunoscuta ca atare de catre blocul „parinte“, exceptia trebuie sa nu fie definita in blocul curent, ci in blocul „parinte“ (sau chiar mai sus in ierarhie), in caz contrar ea putand fi captata de catre blocul „parinte“ doar la categoria *OTHERS.*

Pentru a executa acelasi set de actiuni in cazul mai multor exceptii nominalizate explicit, in sectiunea de prelucrare a exceptiilor se poate utiliza operatorul *OR.*

Pentru a evita tratarea fiecarei erori in parte, se foloseste sectiunea *WHEN OTHERS* care va cuprinde actiuni pentru fiecare exceptie care nu a fost tratata, adica pentru captarea exceptiilor neprevazute sau necunoscute. Aceasta sectiune trebuie utilizata cu atentie deoarece poate masca erori critice sau poate impiedica aplicatia sa raspunda in mod corespunzator.

Propagarea exceptiilor

Daca este declansata o eroare in sectiunea executabila si blocul curent are un *handler* pentru tratarea ei, atunci blocul se termina cu succes, iar controlul este dat blocului imediat exterior.

Daca se produce o exceptie care nu este tratata in blocul curent, atunci exceptia se propaga spre blocul „parinte“, iar blocul *PL/SQL* curent se termina fara succes. Procesul se repeta pana cand fie se gaseste intr-un bloc modalitatea de tratare a erorii, fie se opreste executia si se semnaleaza situatia aparuta ***(****unhandled exception error****).***

Daca este declansata o eroare in partea declarativa a blocului, aceasta este propagata catre blocul imediat exterior, chiar daca exista un *handler* al acesteia in blocul corespunzator sectiunii declarative.

La fel se intampla daca o eroare este declansata in sectiunea de tratare a erorilor. La un moment dat, intr-o sectiune *EXCEPTION****,*** poate fi activa numai o singura exceptie.

Instructiunea *GOTO* nu permite:

- saltul la sectiunea de tratare a unei exceptii;
- saltul de la sectiunea de tratare a unei exceptii, in blocul curent.

Comanda *GOTO* permite totusi saltul de la sectiunea de tratare a unei exceptii la un bloc care include blocul curent.

Exemplu:

Exemplul urmator marcheaza un salt ilegal in blocul curent.

```
DECLARE
v_var NUMBER(10,3);
BEGIN
SELECT dim2/NVL(valoare,0) INTO v var
FROM opera WHERE diml > 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;
```

In continuare, vor fi analizate modalitatile de propagare a exceptiilor in cele trei cazuri comentate: exceptii sesizate in sectiunea declarativa, in sectiunea executabila si in sectiunea de tratare a erorilor.

Exceptie sesizata in sectiunea executabila

Exceptia este sesizata si tratata in subbloc. Dupa 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;
```

Exceptia este sesizata in subbloc, dar nu este tratata in acesta si atunci se propaga spre blocul exterior. Regula poate fi aplicata 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;
```

Exceptie sesizata in sectiunea declarativa

Daca in sectiunea declarativa este generata o exceptie, atunci aceasta se propaga catre blocul exterior, unde are loc tratarea acesteia. Chiar daca exista un *handler* pentru exceptie in blocul curent, acesta nu este executat.

Exemplu:
Sa se realizeze un program prin care sa se exemplifice propagarea erorilor aparute in sectiunea declarativa a unui bloc *PL/SQL*.
Programul calculeaza numarul creatorilor de opere de arta care au lucrari expuse in muzeu.

```
BEGIN
DECLARE
nr_artisti    NUMBER(3) := 'XYZ';
BEGIN
SELECT COUNT (DISTINCT cod_autor)
INTO    nr_artisti
FROM    opera;
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 initializarea variabilei *nr artisti* apare o neconcordanta intre tipul declarat si cel asignat, este generata eroarea interna *VALUE ERROR*. Cum eroarea a aparut in partea declarativa a blocului intern, desi acesta contine un *handler OTHERS* care ar fi putut capta eroarea, *handler-ul* nu este executat, eroarea fiind propagata catre blocul extern unde este tratata in *handler-ul OTHERS* asociat. Aceasta se poate remarca deoarece la executie se obtine mesajul: *„Eroare bloc extern: ORA-06502: PL/SQL: numeric or value error*“

Exceptie sesizata in sectiunea *EXCEPTION*

Daca exceptia este sesizata in sectiunea *EXCEPTION* ea se propaga 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;
```

Informatii despre erori

Pentru a obtine textul corespunzator erorilor la compilare, poate fi utilizata vizualizarea *USERERRORS* din dictionarul datelor. Pentru informatii aditionale referitoare la erori pot fi consultate vizualizarile *ALL ERRORS* sau *DBAERRORS.*

Vizualizarea *USER ERRORS* are campurile:
NAME (numele obiectului),
TYPE (tipul obiectului),
SEQUENCE (numarul secventei),
LINE (numarul liniei din codul sursa in care a aparut eroarea),
POSITION (pozitia in linie unde a aparut eroarea),
TEXT (mesajul asociat erorii).

Exemplu:

Sa se afiseze erorile de compilare din procedura *alfa*.

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

LINE specifica numarul liniei in care apare eroarea, dar acesta nu corespunde liniei efective din fisierul text (se refera la codul sursa depus in *USERSOURCE).*