CUPRINS

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

9. *PL/SQL* – Gestiunea excepţiilor

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

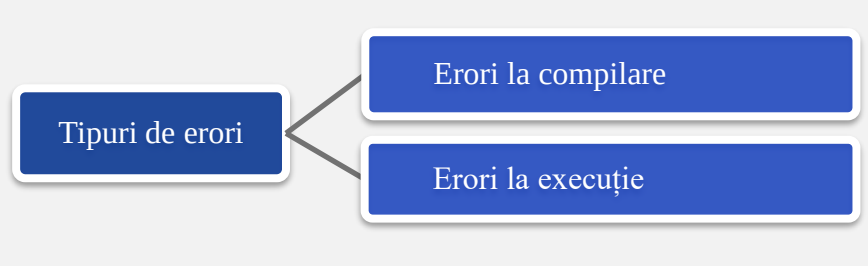

Fig. 9.1. Tipuri de erori

- 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 şi să execute din nou programul;
  - erori la execuţie
    - sunt denumite excepţii;
    - pot apărea datorită deficienţelor de proiectare, defecţiunilor la nivel *hardware*, greşelilor de cod etc.;
    - 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 secţiunea de tratare a excepţiilor, unde va avea loc tratarea erorii;
      - dacă excepţia nu este tratată, atunci aceasta se va propaga în mediul din care a fost lansat programul;
      - nu pot fi anticipate toate excepţiile posibile, dar prin mecanismul de tratare a excepţiilor se poate permite programului să îşi continue execuţia şi în prezenţa anumitor erori.
- Excepţiile pot fi definite şi tratate la nivelul fiecărui bloc din program (bloc 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.

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

- 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 a 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.

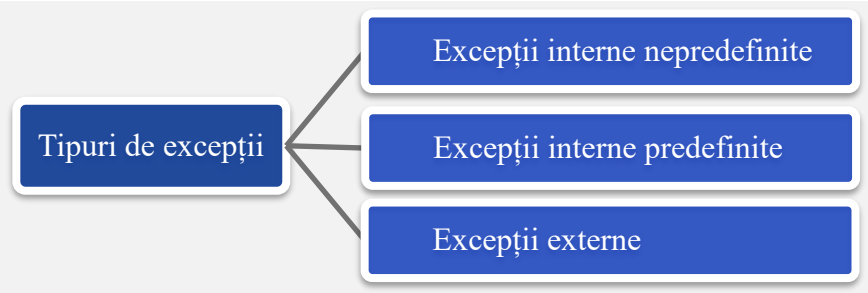

Fig. 9.2. Tipuri de excepţii

- Există următoarele tipuri de excepţii:
  - excepţii interne nepredefinite
    - au un cod de eroare, dar nu au nume asociat decât dacă acesta este precizat de către utilizator;
  - excepţii interne predefinite
    - sunt excepţii interne care au nume asociat de către sistem;
  - excepţii externe
    - sunt excepţii definite de utilizator în blocuri *PL/SQL* anonime, subprograme sau pachete.
- Excepţiile interne:
  - se produc atunci când un bloc *PL/SQL* nu respectă o regulă *Oracle* sau depăşeşte 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*, excepţia asociată acesteia se declanşează implicit.
  - De exemplu, dacă apare eroarea *ORA-01403* (deoarece o comandă *SELECT* nu întoarce nicio linie) atunci implicit *PL/SQL* activează excepţia *NO_DATA_FOUND* (al cărui cod este 100).
  - Cu toate că fiecare astfel de excepţie are asociat un cod specific, ele trebuie referite prin nume.

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

	Excepţii interne nepredefinite	Excepţii interne predefinite	Excepţii externe
Proprietar	Sistemul	Sistemul	Utilizatorul
Cod asociat	Da	Da	Doar dacă utilizatorul îl atribuie
Nume asociat	Doar dacă utilizatorul îl atribuie	Da	Da
Declanşare automată	Da	Da	Nu
Declanşare explicită	Opţional	Opţional	Da

Fig. 9.3. Caracteristicile excepţiilor

9.1. Secţiunea de tratare a excepţiilor

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

```
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;
```

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

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

9.2. Funcţii pentru identificarea excepţiilor

- Funcţia *SQLCODE*:
  - obţine codul excepţiei;
  - întoarce o valoare de tip numeric;
  - codul excepţiei este:
    - un număr negativ, în cazul unei erori interne;
    - 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.
- Funcţia *SQLERRM*:
  - obţine mesajul asociat excepţiei;
  - întoarce un şir de caractere;
  - lungimea maximă a mesajului este de 512 caractere;
  - mesajul asociat excepţiei declanşate poate fi furnizat şi de funcţia *DBMS_UTILITY.FORMAT_ERROR_STACK*.

`Exemplul 9.1` **`- vezi curs`**

9.3. Excepţii interne predefinite

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

Nume excepţie	Cod	Descriere
ACCESS_INTO_NULL	-6530	Asignare de valori atributelor unui obiect neiniţializat.
CASE_NOT_FOUND	-6592	Nu este selectată nici una din clauzele *WHEN* ale lui *CASE* şi nu există nici clauza *ELSE* .
COLLECTION_IS_NULL	-6531	Aplicarea unei metode (diferite de *EXISTS*) unui tabel imbricat sau unui vector neiniţializat.

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

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	Operaţie ilegală asupra unui cursor.
INVALID_NUMBER	-1722	Conversie nepermisă de la tipul şir de caractere la număr.
LOGIN_DENIED	-1017	Nume sau parolă incorecte.
NO_DATA_FOUND	+100	Comanda *SELECT* nu întoarce nicio linie.
NOT_LOGGED_ON	-1012	Programul *PL/SQL* apelează baza de date fără să fie conectat la *Oracle*.
PROGRAM_ERROR	-6501	*PL/SQL* are o problemă internă.
ROWTYPE_MISMATCH	-6504	Incompatibilitate între parametrii actuali şi formali, la deschiderea unui cursor parametrizat.
SELF_IS_NULL	-30625	Apelul unei metode când instanţa este *NULL*.
STORAGE_ERROR	-6500	*PL/SQL* are probleme cu spaţiul de memorie.
SUBSCRIPT_BEYOND_ COUNT	-6533	Referire la o componentă a unui tablou imbricat sau vector, folosind un index mai mare decât numărul elementelor colecţiei 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 şir de caractere într-un *ROWID* nu se poate face deoarece şirul nu reprezintă un *ROWID* valid.
TIMEOUT_ON_ RESOURCE	-51	Expirarea timpului de aşteptare pentru eliberarea unei resurse.
TRANSACTION_BACKED _OUT	-61	Tranzacţia este anulată datorită unei interblocări.
TOO_MANY_ROWS	-1422	Comanda *SELECT* întoarce mai multe linii.

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

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

- 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.

`Exemplul 9.2` **`– vezi curs`**

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

`Exemplul 9.3` **`– vezi curs`**

9.4. Excepţii interne nepredefinite

- Excepţiile interne nepredefinite se declară în secţiunea declarativă a blocului *PL/SQL* şi sunt declanşate implicit de către *server*-ul *Oracle*.
- Diferenţierea acestor erori este posibilă doar cu ajutorul codului asociat lor.
- Există două metode de tratare a excepţiilor interne nepredefinite, folosind:
  - clauza *WHEN OTHERS* din secţiunea *EXCEPTION* a blocului;
  - directiva de compilare (pseudo-instrucţiune) *PRAGMA EXCEPTION_INIT*.
- Directiva *PRAGMA EXCEPTION_INIT*
  - permite asocierea unui nume pentru o excepţie al cărui cod de eroare intern este specificat;
  - având un nume specificat pentru excepţie, permite referirea acesteia în secţiunea *EXCEPTION* a blocului;
  - este procesată în momentul compilării (nu la execuţie);

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

- trebuie să apară în partea declarativă a unui bloc, pachet sau subprogram, după definirea numelui excepţiei;
- poate să apară de mai multe ori într-un program; de asemenea, pot fi asignate mai multe nume pentru acelaşi cod de eroare.

- Pentru a trata o eroare folosind directiva *PRAGMA EXCEPTION_INIT* trebuie urmaţi paşii de mai jos:

1) se declară numele excepţiei în partea declarativă a blocului:

```
nume_excepţie EXCEPTION;
```

2) se asociază numele excepţiei cu un cod de eroare standard *Oracle*:

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

3) se referă excepţia în secţiunea *EXCEPTION* a blocului (excepţia este tratată automat, fără a fi necesară comanda *RAISE*):

```
WHEN nume_excepţie THEN set_de_instrucţiuni
```

Exemplul 9.4 **– vezi curs**

9.5. Excepţii externe

- Excepţiile externe:
  - sunt excepţii definite de utilizator;
  - sunt declarate în secţiunea declarativă a unui bloc, subprogram sau pachet;
  - sunt activate explicit în partea executabilă a blocului (folosind comanda *RAISE* însoţită de numele excepţiei);
  - pot să apară în toate secţiunile unui bloc, subprogram sau pachet;
  - nu pot să apară în instrucţiuni de atribuire sau în comenzi *SQL*;
  - în mod implicit, au asociat acelaşi cod (+1) şi acelaşi mesaj (*User-Defined Exception*).

- Sintaxa de declarare şi prelucrare a excepţiilor externe :

```
DECLARE
     nume_excepţie  EXCEPTION; -- declarare excepţie
BEGIN
      …
     RAISE  nume_excepţie;      -- activare excepţie
      -- execuţia este întreruptă şi se transferă
      -- controlul în secţiunea EXCEPTION
```

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

```
        …
    EXCEPTION
        WHEN nume_excepţie THEN
        -- definire mod de tratare a erorii
        …
    END;
```

Exemplul 9.5 – vezi curs

- Activarea unei excepţii externe poate fi făcută şi cu ajutorul procedurii *RAISE_APPLICATION_ERROR*, furnizată de pachetul *DBMS_STANDARD*.
  - Această procedură poate fi utilizată pentru a întoarce un mesaj de eroare unităţii 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 şi –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 aplicaţie poate apela procedura *RAISE_APPLICATION_ERROR* numai dintr-un subprogram stocat (sau metodă).
- ❖ Dacă procedura *RAISE_APPLICATION_ERROR* este apelată, atunci subprogramul se termină şi sunt întoarce codul şi mesajul asociate erorii respective.

- Procedura *RAISE_APPLICATION_ERROR* poate fi apelată în secţiunea executabilă, în secţiunea de tratare a excepțiilor sau simultan în ambele secţiuni.
  - În secţiunea executabilă:

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

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

- În secţiunea de tratare a excepţiilor:

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

- În ambele secţiuni:

```
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 comunicaţia dintre *client* şi *server*, transmiţând aplicaţiei *client* erori specifice aplicaţiei de pe *server* (de obicei, un *trigger*).
- Procedura *RAISE_APPLICATION_ERROR* este doar un mecanism folosit pentru comunicaţia dintre *server* şi *client* a unei erori definite de utilizator, care permite ca procesul *client* să trateze excepţia.

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);
```

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

```
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 excepţiilor

- Dacă se declanşează o excepţie 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 doreşte totuşi continuarea prelucrării datelor, este suficient ca instrucţiunea care a declanşat excepţia să fie inclusă într-un subbloc.
- ❖ În acest caz, după tratarea excepției și ieșirea din subbloc, se continuă secvenţa de instrucţiuni 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.

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

```
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 secţiune pentru tratarea erorilor unui bloc.

  Variante de rezolvare a acestor situații:

  - utilizarea unui contor care să identifice instrucţiunea *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 instrucţiuni *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;
```

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

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

9.7. Activarea excepţiilor

- Există două metode de activare a unei excepţii:
  - activarea explicită a excepţiei (definită de utilizator sau predefinită) în interiorul blocului, cu ajutorul comenzii *RAISE*;
  - activarea automată a excepţiei asociate unei erori *Oracle*.
- Excepţiile pot fi generate în oricare dintre secțiunile unui bloc *PL/SQL* (în secțiunea declarativă, în secţiunea executabilă sau în secţiunea 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 tratare a excepţiilor se poate utiliza operatorul *OR* (*WHEN excepție_1 OR excepție_2 THEN ...*).
- 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.

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

9.8. Propagarea excepţiilor

- Dacă este declanşată o eroare în secţiunea 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 excepţia se propagă spre blocul „părinte“, iar blocul 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 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 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.

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;
/
```

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

9.8.1 Excepţie declanșată în secţiunea executabilă

- Excepţia este produsă şi 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;
```

- Excepţia 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 Excepţie declanșată 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.

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

Exemplul 9.8

```
BEGIN
  DECLARE
    nr_produse   NUMBER(10) := ' ';
    -- este generata eroarea VALUE_ERROR
  BEGIN
    SELECT  COUNT (DISTINCT id_produs)
    INTO    nr_produse
    FROM    facturi_produse;
  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 Excepţie declanșată în secţiunea *EXCEPTION*

- Dacă excepţia este declanșată în secţiunea *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;
```

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

9.9. Informaţii despre erori

- Pentru a obţine textul corespunzător erorilor apărute 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* 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 fişierul 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"
```

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.

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 şi gestiunea tranzacţiilor*, Editura Albastră, Cluj-Napoca, 2004
3. Oracle and/or its affiliates, *Oracle Database Concepts*, 1993, 2017
4. Oracle and/or its affiliates, *Oracle Database Performance Tuning Guide*, 2013, 2017
5. Oracle and/or its affiliates, *Oracle Database SQL Language Reference*, 1996, 2017
6. Oracle and/or its affiliates, *Oracle Database PL/SQL Language Reference*, 1996, 2017
7. Oracle and/or its affiliates, *Oracle Database Administrator's Guide*, 2001, 2010
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

Copyright © 2012-2019 Lect. Dr. Gabriela Mihai. Toate drepturile rezervate.