

Limbajul PL/SQL- Obiecte

- Tratarea obiectelor neinitializate

In expresii attributele unui obiect neinitializat sunt evaluate la NULL.

Asignarea de valori pentru un attributele unui obiect neinitializat produce exceptia ACCESS_INTO_NULL.

Operatorul IS NULL aplicat obiectului neinitializat sau atributelor acestuia da TRUE.

```
DECLARE
```

```
  rat Rational; -- rat este atomic null
```

```
BEGIN
```

```
  IF rat IS NULL THEN ... --expresia este TRUE
```

```
  IF rat.numarator IS NULL THEN ...-- TRUE
```

```
  rat:=Rational(NULL,NULL); -- initializeaza rat
```

```
  rat.numarator:=10; -- nu produce exceptie
```

```
  rat:=NULL; -- rat devine atomic nul
```

```
  rat.numarator:=20; -- produce exceptia ACCESS_INTO_NULL
```

```
EXCEPTION WHEN ACCESS_INTO_NULL THEN .. END;
```

Limbajul PL/SQL- Obiecte

- Definitii de tip forward

```
CREATE TYPE Emp AS OBJECT (  
    nume VARCHAR2(15),  
    dept REF departament,    --incorect  
    ...  
);
```

```
CREATE TYPE departament AS OBJECT (  
    numar INTEGER,  
    manager Emp, ... );
```

```
CREATE TYPE departament; -- definitie forward inainte de Emp  
departament – tip incomplet
```

tip obiect incomplet are attribute dar se refera la tipuri nedefinite

```
CREATE TABLE nume_tabela OF tip_object -- tabela de obiecte
```

O linie in tabela de obiecte are un identificator de obiect si serveste ca referinta la acel obiect

Limbajul PL/SQL- Obiecte

- Selectarea obiectelor

```
CREATE TYPE persoana AS OBJECT (  
    nume VARCHAR2(15),  
    prenume VARCHAR2(15),  
    data_nasterii DATE,  
    adresa Address,  
    telefon VARCHAR2(15) )
```

/

```
CREATE TABLE persoane OF persoana
```

/

```
BEGIN
```

```
    INSERT INTO emp
```

```
        SELECT * FROM persoane p
```

```
        WHERE p.nume LIKE 'A%';
```

Limbajul PL/SQL- Obiecte

VALUE returneaza valoarea unui obiect

BEGIN

INSERT INTO emp

SELECT VALUE(p) FROM persoane p

WHERE p.numa LIKE 'A%';

DECLARE

p1 persoane;

p2 persoane;

BEGIN

SELECT VALUE(p) INTO p1 FROM persoane p

WHERE p.numa='IONESCU';

p2:=p1; p1.numa:=expresie;

... END;

Limbajul PL/SQL- Obiecte

- Operatorul REF

Variabila corelata= variabila de linie sau alias pentru o tabela de obiecte
REF are ca argument o variabila corelata si returneaza referinta.

```
BEGIN
```

```
INSERT INTO persoane_ref
```

```
SELECT REF(p) FROM persoane p
```

```
WHERE p.nume LIKE 'A%';
```

```
DECLARE
```

```
p_ref REF persoana;
```

```
telefon VARCHAR2(15);
```

```
BEGIN
```

```
SELECT REF(p),p.telefon INTO p_ref, telefon
```

```
FROM persoane p
```

```
WHERE p.nume='IONESCU'; -- trebuie o singura linie
```

```
END;
```

Limbajul PL/SQL- Obiecte

```
DECLARE
```

```
  p_ref    REF persoana;
```

```
  wnume   VARCHAR2(15);
```

```
BEGIN
```

```
  SELECT REF(p) INTO p_ref FROM persoane p
     WHERE p.nume=wnume;
```

```
  UPDATE persoane p
```

```
     SET p= persoana('ION','ION',...)
```

```
     WHERE REF(p)= p_ref;
```

```
END;
```

Referinte dangling (pointeaza la un obiect inexistent), IS DANGLING.

```
BEGIN
```

```
  UPDATE departament SET manager=NULL WHERE manager IS
  DANGLING;
```

Limbajul PL/SQL- Obiecte

- Operatorul Deref

argumentul =referinta la obiect, returneaza valoarea (obiectul) la care pointeaza. Daca ref este dangling, atunci se returneaza obiectul nul.

```
DECLARE
```

```
p1 persoana;
```

```
p1_ref REF persoana;
```

```
wnume VARCHAR2(15);
```

```
BEGIN
```

```
SELECT REF(p) INTO p1_ref FROM persoane p WHERE p.marca=10;
```

```
SELECT Deref(p1_ref) INTO p1 FROM DUAL;
```

```
wnume:=p1.nume;
```

```
DBMS_OUTPUT.PUT_LINE('Numele este; '||wnume);
```

```
END;
```

Limbajul PL/SQL- Obiecte

Operatorul Deref poate fi folosit in dereferinte succesive

```
CREATE TYPE persoanan_ref AS OBJECT (p_ref REF persoana)
```

```
/
```

```
CREATE TABLE persoanen_ref OF persoanan_ref
```

```
/
```

```
DECLARE
```

```
  wnume VARCHAR2(15);
```

```
  obiect_1 persoanan_ref :=persoanan_ref(NULL);
```

```
  ref_ref_p REF persoanan_ref;
```

```
  p persoana;
```

```
  ref_p REF persoana;
```

```
  ref_p1 persoanan_ref;
```

```
BEGIN
```

Limbajul PL/SQL- Obiecte

```
SELECT REF(a) INTO ref_p FROM persoane a WHERE  
a.marca=10;
```

```
obiect_1.p_ref:=ref_p;
```

```
INSERT INTO persoanen_ref VALUES(obiect_1);
```

```
SELECT REF(c) INTO ref_ref_p FROM persoanen_ref c  
WHERE c.p_ref=ref_p;
```

```
SELECT Deref(ref_ref_p) INTO ref_p1 FROM DUAL;
```

```
SELECT Deref(ref_p1.p_ref) INTO p FROM DUAL;
```

```
wnume:=p.num;
```

```
DBMS_OUTPUT.PUT_LINE(wnume);
```

```
END;
```

```
/
```

Limbajul PL/SQL- Obiecte

- Operatorul Deref nu poate fi folosit in instructiuni procedurale:

```
P1:=Deref(p_ref); --incorect
```

In comenzi SQL folosim notatia . Pentru a naviga de la coloane obiect la attribute REF si de la attribute REF la alt atribut REF.

```
alias_tabela.coloana_obiect.atribut_ref
```

```
alias_tabela.coloana_obiect.atribut_ref.atribut
```

```
alias_tabela.coloana_ref.atribut
```

```
CREATE TYPE Address AS OBJECT (
```

```
    strada VARCHAR2(15),
```

```
    oras   VARCHAR2(15),
```

```
    cod_postal INTEGER)
```

```
/
```

```
CREATE TYPE Person AS OBJECT (nume VARCHAR2(15),prenume
```

```
VARCHAR2(15),adresa_dom REF Address,telefon VARCHAR2(15),strada  
    VARCHAR2(15))
```

```
/      CREATE TABLE persons OF Person
```

```
/
```

Limbajul PL/SQL- Obiecte

DECLARE

addr1 Address;

addr2 Address;

wstrada VARCHAR2(15);

BEGIN

```
SELECT Deref(adresa_dom) INTO addr1 FROM persons p
      WHERE p.nume='ION';
```

```
SELECT p.adresa_dom.strada INTO wstrada FROM persons p
      WHERE p.prenume='ioana';
```

- Inserarea de obiecte

```
INSERT INTO persons VALUES(Person('ION','ioana',...));
```

Utilizam clauza RETURNING sa memoram referinte in variabile locale

Limbajul PL/SQL- Obiecte

DECLARE

p1_ref REF person;

p2_ref REF person;

BEGIN

INSERT INTO persons p

VALUES(Person('IONESCU','dorin',...))

RETURNING REF(p) INTO p1_ref;

INSERT INTO persons1

SELECT VALUE(p) FROM persons p

WHERE p.prenume LIKE 'A%';

UPDATE persons p SET adresa_dom='Iasi....' WHERE...;

DELETE FROM persons p WHERE ...;

Limbajul PL/SQL- Obiecte

- DECLARE

```
p_ref REF Person;
```

```
wprenume VARCHAR2(15);
```

```
BEGIN
```

```
wprenume:='bill';
```

```
SELECT REF(p) INTO p_ref FROM persons p
```

```
WHERE p.prenume = wprenume;
```

```
UPDATE persons p SET p = Person('Jill', 'Anders', '11-NOV-67', ...)
```

```
WHERE REF(p) = p_ref;
```

```
END;
```

```
/
```

Limbajul PL/SQL- Obiecte

- Mostenirea tipurilor

```
CREATE TYPE T1 AS OBJECT (...) NOT FINAL;
```

```
CREATE TYPE T2 UNDER T1(...) {NOT FINAL|FINAL};
```

T1- supertip(parinte) pentru T2, iar T2 subtip(fiu)al lui T1

FINAL- nu se pot deriva subtipuri din el

Subtipul contine toate attributele si metodele tipului parinte si poate contine attribute si metode suplimentare care pot supraincarca metodele tipului parinte.

FINAL MEMBER FUNCTION f (...)- in tipul parinte implica imposibilitatea supraincarcarii lui f in orice subtip al celui curent.

```
CREATE TYPE tip_persoana AS OBJECT (  
  nume VARCHAR2(20),cnp CHAR(13),adresa VARCHAR2(25))  
NOT FINAL;
```

```
CREATE TYPE tip_student UNDER tip_persoana (cod_fac NUMBER) NOT FINAL;
```

```
CREATE TYPE tip_functionar UNDER tip_persoana(marca NUMBER);
```

```
CREATE TYPE student1 UNDER tip_student (bursa NUMBER);
```

Limbajul PL/SQL- Obiecte

```
CREATE TYPE T AS OBJECT (... ,MEMBER PROCEDURE P1, FINAL MEMBER  
FUNCTION f1(...)) NOT FINAL;
```

```
CREATE TYPE tip_adresa AS OBJECT(...) NOT INSTANTIABLE  
NOT FINAL;  
CREATE TYPE tip_adresa1 UNDER tip_adresa(...);  
CREATE TYPE tip_adresa2 UNDER tip_adresa(...);
```

```
CREATE TABLE person_v OF tip_persoana
```

- Operatorul TREAT – returneaza obiecte de un tip specificat
- SELECT TREAT(REF(p) AS REF tip_student);

```
FROM Person_v p; --returneaza referintele persoanelor ce sunt --instante ale  
subtipului tip_student
```

```
SELECT REF(p) FROM Person_v p WHERE VALUE(p) IS OF (tip_functionar,  
tip_student) ;
```

```
SELECT REF(p) FROM Person_v p WHERE VALUE(p) IS OF(ONLY  
tip_student);
```

Limbajul PL/SQL- Obiecte

- Constructor definit de utilizator

```
CONSTRUCTOR FUNCTION tip_obiect( lista) RETURN SELF AS  
RESULT
```

```
CREATE OR REPLACE TYPE paralelipiped AS OBJECT (
```

```
-- Tipul are 3 attribute
```

```
lungime NUMBER, latime NUMBER, inaltime NUMBER, volum  
NUMBER,
```

```
-- Definim un constructor cu 3 parametri.
```

```
CONSTRUCTOR FUNCTION paralelipiped (plungime NUMBER,  
platime NUMBER,pinaltime NUMBER) RETURN SELF AS  
RESULT );
```

```
/
```

```
CREATE OR REPLACE TYPE BODY paralelipiped AS
```

```
CONSTRUCTOR FUNCTION paralelipiped (plungime NUMBER,  
platime NUMBER,pinaltime NUMBER) RETURN SELF AS  
RESULT
```

Limbajul PL/SQL- Obiecte

AS

BEGIN

SELF.lungime := plungime;

SELF.latime:= platime;

SELF.inaltime:=pinaltime;

SELF.volum:=plungime*platime*pinaltime;

RETURN;

END;

END;

/

DECLARE

par1 paralelipiped; par2 paralelipiped;

BEGIN

par1:=NEW paralelipiped(10,20,30,6000);

par2:= NEW paralelipiped(10,20,30);

Limbajul PL/SQL- Obiecte

- Parametrul SELF

Fiecare metoda are primul parametru cu numele SELF. In mod implicit orice atribut necalificat in functie membru sau procedura membru este calificat prin SELF.

```
CREATE OR REPLACE TYPE cladire AS OBJECT (  
    Numecladire      VARCHAR2(40),  
    Adresacladire   address,  
    Managercladire  INTEGER,  
    MEMBER PROCEDURE SchimbareManager (SELF IN OUT  
    cladire, NouManager IN INTEGER),  
    ORDER MEMBER FUNCTION Comparare (SELF IN cladire,  
    altacladire IN cladire) RETURN INTEGER  
)  
/
```

Limbajul PL/SQL- Obiecte

```
CREATE OR REPLACE TYPE BODY cladire AS
MEMBER PROCEDURE SchimbareManager(SELF IN OUT cladire,
    NouManager IN INTEGER) IS
BEGIN
    SELF.Managercladire:= NouManager;
END;
ORDER MEMBER FUNCTION Comparare (SELF IN cladire,
    altacladire IN cladire) RETURN INTEGER IS
    Numecladire1    VARCHAR2(40);
    Numecladire2    cladire.Numecladire%TYPE; BEGIN
Numecladire1 := upper(ltrim(rtrim(SELF.Numecladire)));
Numecladire2:= upper(ltrim(rtrim(altacladire.Numecladire)));
IF Numecladire1 = Numecladire2 THEN RETURN 0;
    ELSIF Numecladire1 < Numecladire2 THEN    RETURN -1;
ELSE    RETURN 1;    END IF;    END;    END;
```

Limbajul PL/SQL- Tratarea erorilor

- Tratarea erorilor

In PL/SQL o eroare este numita exceptie. Exceptiile pot fi predefinite(interne) sau definite de utilizator. Exemple de exceptii predefinite: *division by zero, out of memory* . ZERO_DIVIDE and STORAGE_ERROR. Exceptiile interne sunt activate automat.

Exceptiile utilizator sunt activate de instructiuni RAISE.

```
SET SERVEROUTPUT ON;
```

```
DECLARE pret_stoc NUMBER := 9.73;
```

```
Pret_vinzare NUMBER := 0;
```

```
Coeficient NUMBER;
```

```
BEGIN
```

```
    coeficient:= pret_stoc / pret_vinzare;
```

```
    dbms_output.put_line('Pretul stoc/pret vinzare = ' ||coeficient);
```

```
EXCEPTION – prelucrarea exceptiilor
```

```
    BEGIN
```

Limbajul PL/SQL- Tratarea erorilor

```
WHEN ZERO_DIVIDE THEN --tratarea impartirii prin zero
    dbms_output.put_line('Pretul de vinzare este zero');
    coeficient:= NULL;
```

```
WHEN OTHERS THEN -- tratarea altor erori
    dbms_output.put_line('Alte tipuri de erori:');
    coeficient := null;
```

```
END;
```

```
END;
```

```
/
```

```
BEGIN
```

```
    coeficient := CASE pret_vinzare
```

```
        WHEN 0 THEN NULL ELSE pret_stoc/ pret_vinzare
```

```
        END;
```

```
END;
```

```
/
```

Limbajul PL/SQL- Tratarea erorilor

- Avantajele exceptiilor

Fara rutina de prelucrare a exceptiilor, cind se executa o comanda trebuie sa vedem daca nu au aparut situatii de eroare:

```
BEGIN
```

```
SELECT ... INTO ...
```

```
--verificarea daca nu a aparut eroare
```

```
SELECT ... INTO...
```

```
--verificarea daca nu a aparut eroare
```

```
END;
```

```
BEGIN
```

```
SELECT ...
```

```
SELECT ...
```

```
SELECT ...
```

```
EXCEPTION
```

```
WHEN NO_DATA_FOUND THEN --tratarea tuturor erorilor 'no data  
found'
```

Limbajul PL/SQL- Tratarea erorilor

Exception	Oracle Error	SQLCODE	Value
1.ACCESS_INTO_NULL	ORA-06530		-6530
2.CASE_NOT_FOUND	ORA-06592		-6592
3.COLLECTION_IS_NULL	ORA-06531		-6531
4.CURSOR_ALREADY_OPEN	ORA-06511		-6511
5.DUP_VAL_ON_INDEX	ORA-00001		-1
6.INVALID_CURSOR	ORA-01001		-1001
7.INVALID_NUMBER	ORA-01722		-1722
8.LOGIN_DENIED	ORA-01017		- 1017
9.NO_DATA_FOUND	ORA-01403		+100
10.NOT_LOGGED_ON	ORA-01012		-1012
11.PROGRAM_ERROR	ORA-06501		-6501
12.ROWTYPE_MISMATCH	ORA-06504		- 6504
13.SELF_IS_NULL	ORA-30625		-30625
14.STORAGE_ERROR	ORA-06500	194	-6500

Limbajul PL/SQL- Tratarea erorilor

15.SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
16.SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
17.SYS_INVALID_ROWID	ORA-01410	-1410
18.TIMEOUT_ON_RESOURCE	ORA-00051	-51
19.TOO_MANY_ROWS	ORA-01422	-1422
20.VALUE_ERROR	ORA-06502	-6502
21.ZERO_DIVIDE	ORA-01476	-1476

- 1.Programul asigneaza valori la un atribut al unui obiect neinitializat.
- 2.Nu s-a selectat niciun WHEN din CASE si nu exista clauza ELSE.
- 3.Programul aplica o metoda(alta decit EXISTS) la o colectie neinitializata sau se asigneaza valori la elementele colectiei neinitializata.
- 4.Programul incearca sa deschida un cursor deja deschis. Un cursor FOR LOOP este deschis automat.

Limbajul PL/SQL- Tratarea erorilor

5. Programul incearca sa memoreze valori duplicate ale cimpului cu restrictia PRIMARY KEY sau UNIQUE KEY.
6. Programul incearca o operatie nepermisa pe un cursor, de ex. Inchiderea unui cursor nedeschis.
7. Conversia unui sir de caractere la un numar esueaza, deoarece sirul nu are numai caractere numerice (pentru instr. SQL). Pentru instructiuni procedurale eroarea: VALUE_ERROR. Exceptia apare si cind expresia LIMIT din FETCH cu BULK COLLECTION nu se evalueaza la un numar pozitiv.
8. Programul incearca o logare la Oracle cu nume sau password gresit.
9. Instr. SELECT INTO nu returneaza nicio linie sau programul refera un element sters dintr-o tabela nested sau un element neinitializat dintr-o tabela index by.
10. Programul face o chemare de baza de date fara a fi conectat la Oracle.
11. PL/SQL are o problema interna.

Limbajul PL/SQL- Tratarea erorilor

12. Variabila cursor intr-o asignare returneaza valori in variabile de tipuri diferite de elementele cursorului.
13. Programul incearca chemarea unei metode cu un obiect neinitializat.
14. PL/SQL executa in afara memoriei sau memoria a fost corupta.
15. Programul face referinta la un element al unei colectii folosind un indice mai mare decat numarul elementelor colectiei.
16. Programul face referinta la un element al unei colectii folosind un indice in afara domeniului declarat pentru elementele colectiei.
17. Conversia unui sir de caractere la ROWID esueaza deoarece sirul nu este un ROWID corect.
18. S-a depasit timpul de cind Oracle asteapta o resursa.
19. O instructiune SELECT INTO returneaza cel putin 2 linii.
20. A aparut o eroare in evaluarea unei expresii aritmetice, de conversie, de trunchiere sau din cauza unei restrictii de dimensiune.
21. Programul incearca o impartire prin zero.

Limbajul PL/SQL- Tratarea erorilor

- Definirea exceptiilor utilizator
 - Declararea in partea declarativa a unui bloc PL/SQL, subprogram sau pachet:

```
DECLARE
```

```
    nume_exceptie EXCEPTION;
```

- nume_exceptie nu poate sa apara in asignari sau instructiuni SQL.
- nu se poate declara o exceptie de 2 ori in acelasi bloc, dar aceeasi exceptie poate sa apara in blocuri diferite.
- exceptiile declarate intr-un bloc sunt locale acelui bloc si globale pentru toate subblocurile acelui bloc.
- daca redeclaram exceptia globala intr-un subbloc, atunci cea locala are prioritate, subblocul va putea referi exceptia globala in forma :

```
    nume_bloc.nume_exceptie
```

```
DECLARE excep1 EXCEPTION;
```

```
    nrcont NUMBER;
```

```
BEGIN
```

Limbajul PL/SQL- Tratarea erorilor

...

```
DECLARE --inceput subbloc
```

```
  excep1 EXCEPTION; --are prioritate
```

```
  nrcont NUMBER;
```

```
  BEGIN
```

...

```
  IF ... THEN
```

```
    RAISE excep1; -- nu este prelucrata
```

```
  END IF;
```

```
  END; -sfirsitul subblocului
```

```
EXCEPTION
```

```
WHEN excep1 THEN... --nu este prelucrata exceptia lansata de RAISE
```

...

```
END; - -RAISE si WHEN se refera la nume diferite
```

Limbajul PL/SQL- Tratarea erorilor

- Pragma EXCEPTION_INIT

Pentru prelucrarea exceptiilor interne folosim WHEN OTHERS.. Sau pragma EXCEPTION_INIT. O pragma este o directiva de compilare.

In PL/SQL pragma EXCEPTION_INIT asociaza un nume_exceptie cu un numar de eroare Oracle, ceea ce permite referirea la exceptie prin nume.

```
DECLARE
```

```
nume_exceptie EXCEPTION;
```

```
PRAGMA EXCEPTION_INIT(nume_exceptie,-50);
```

```
BEGIN
```

```
...
```

```
EXCEPTION
```

```
WHEN nume_exceptie THEN
```

```
--prelucrarea erorii
```

```
END;
```

Limbajul PL/SQL- Tratarea erorilor

- Definirea propriilor mesaje de eroare

```
RAISE_APPLICATION_ERROR(numar_eroare,mesaj  
                        [{TRUE|FALSE}]);  
numar_eroare∈[-20000,-20999], lungime(mesaj)≤2048.
```

Apelul se realizeaza numai dintr-un subprogram memorat.

Apelul ei determina terminarea subprogramului si returnarea unui numar de eroare si mesaj.

```
CREATE PROCEDURE crestere_salariu (emp_id NUMBER,rata  
NUMBER) AS  
    salariu_actual NUMBER;  
BEGIN  
    SELECT sal INTO salariu_actual FRM emp WHERE empno=emp_id;  
    IF salariu_actual IS NULL THEN  
RAISE_APPLICATION_ERROR(-20100,'Lipseste salariul);  
    ELSE
```

Limbajul PL/SQL- Tratarea erorilor

```
UPDATE emp SET sal=salariu_actual+rata WHERE empno=emp_id;  
END IF;  
END crestere_salariu;
```

/

- Lansarea exceptiilor-RAISE

```
DECLARE
```

```
exceptie_stoc EXCEPTION;
```

```
cantmin_stoc NUMBER(5);
```

```
BEGIN
```

```
...
```

```
IF cantmin_stoc <1000 THEN
```

```
    RAISE exceptie_stoc;
```

```
END IF;
```

```
EXCEPTION WHEN exceptie_stoc THEN
```

```
--prelucreaza eroarea    END;
```

Limbajul PL/SQL- Tratarea erorilor

- Putem lansa explicit erorile predefinite prin: RAISE nume_exceptie:

```
DECLARE
```

```
  nr1 INTEGER;
```

```
BEGIN
```

```
  IF nr1 NOT IN (1,2,3,4) THEN
```

```
    RAISE INVALID_NUMBER;
```

```
  END IF;
```

```
  ...
```

```
EXCEPTION
```

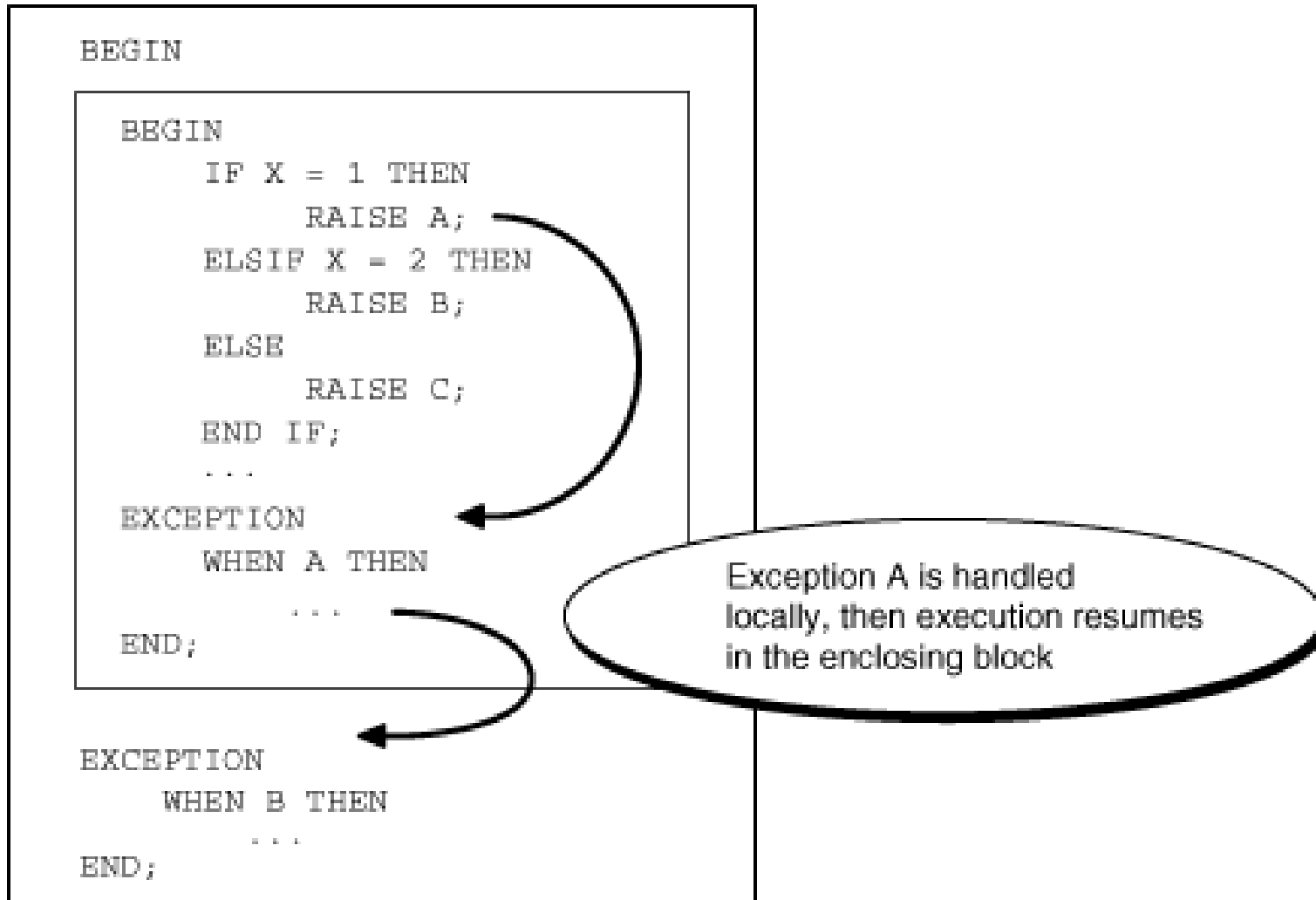
```
  WHEN INVALID_NUMBER THEN
```

```
    ROLLBACK;
```

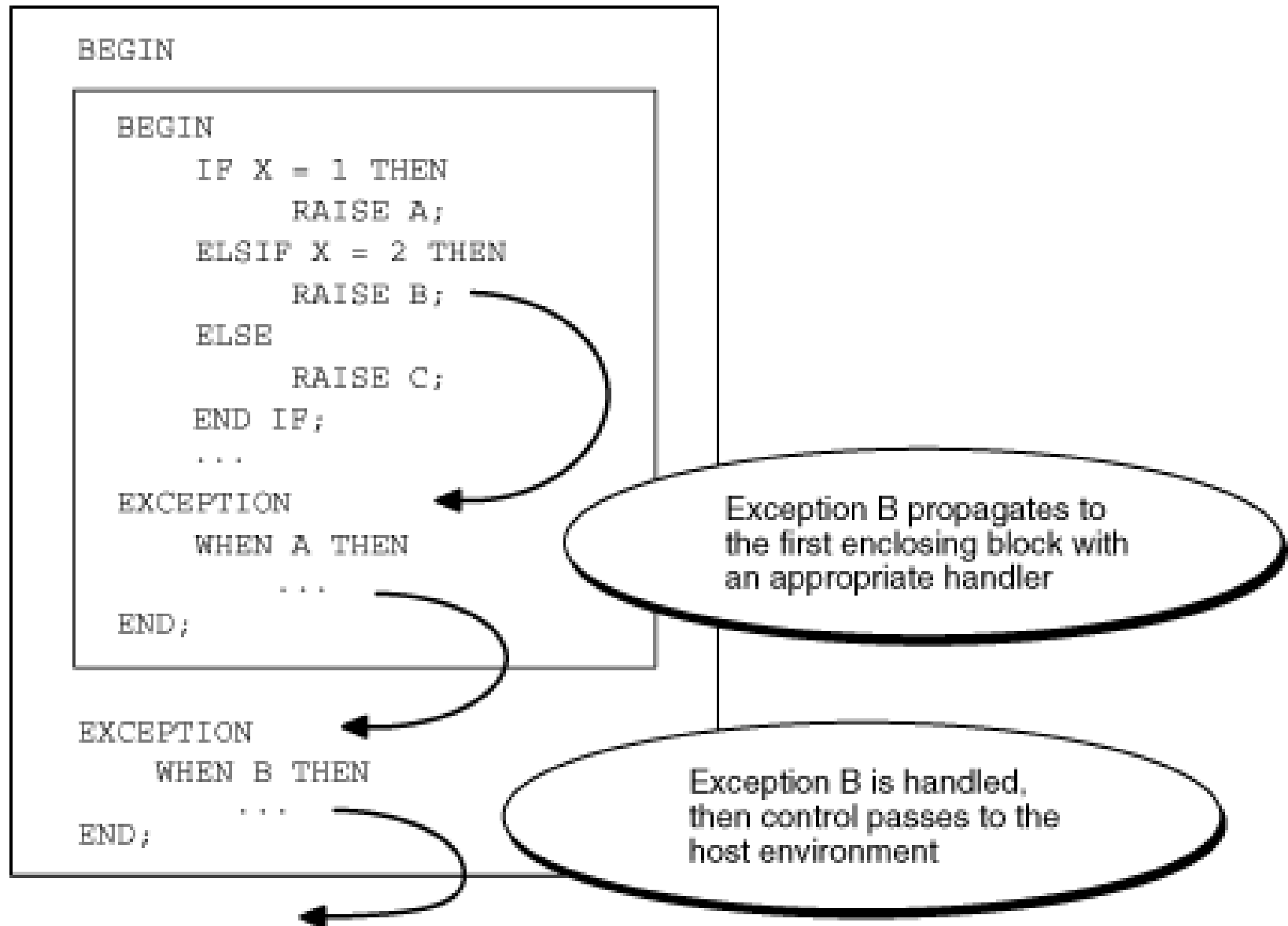
```
  ...
```

```
END;
```

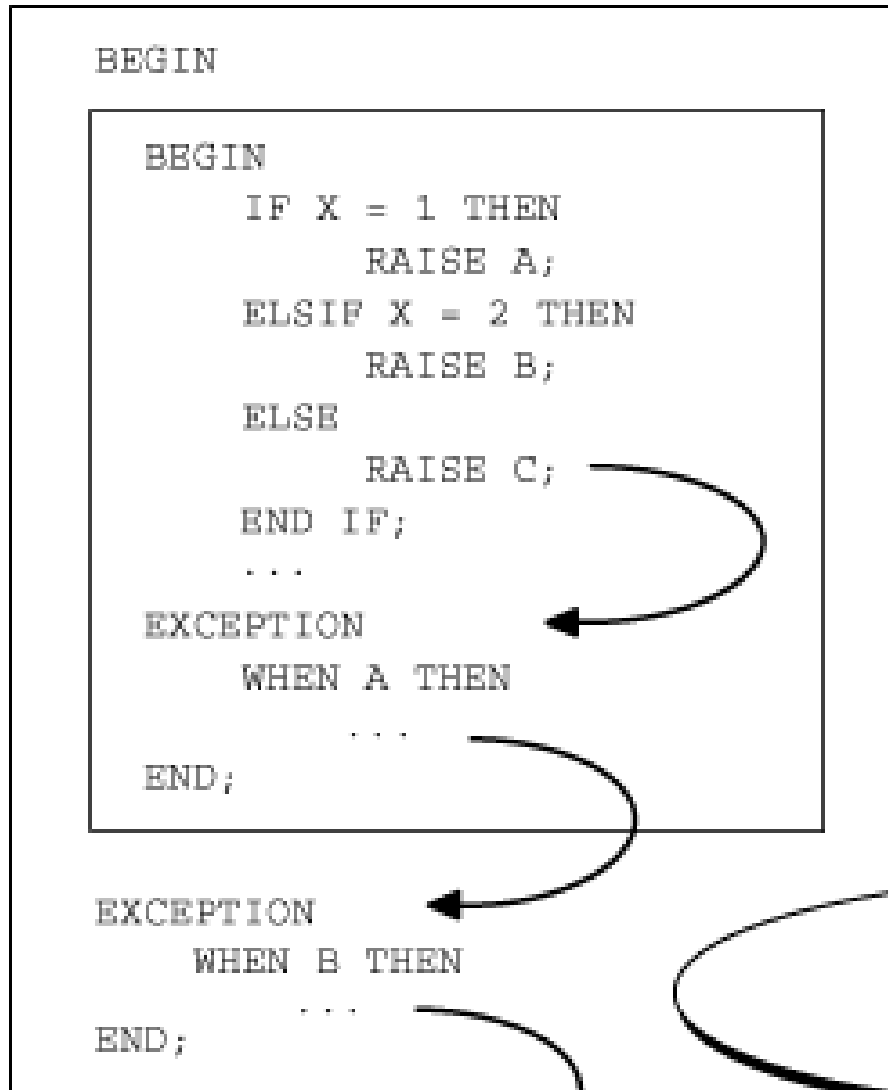
- Propagarea exceptiilor



Limbajul PL/SQL- Tratarea erorilor



Limbajul PL/SQL- Tratarea erorilor



Exception C has no handler, so an unhandled exception is returned to the host environment

Limbajul PL/SQL- Tratarea erorilor

Nu putem avea:

```
EXCEPTION
```

```
    WHEN nume_exceptie....
```

unde nume_exceptie este definit intr-un subbloc al blocului curent.

Totusi daca folosim WHEN OTHERS THEN... exceptia definita intr-un subbloc se propaga in blocul inclus.

```
BEGIN
```

```
...
```

```
DECLARE
```

```
    except1 EXCEPTION;
```

```
    BEGIN
```

```
        IF ... THEN RAISE except1; END IF;
```

```
    END;
```

```
EXCEPTION
```

```
WHEN OTHERS THEN ROLLBACK;      END;
```