

Sistemi baza podataka

Slavica Aleksić
slavica@uns.ac.rs

Kursorska FOR petlja

```
FOR record_var IN naziv_kursora  
  [(lista_stvarnih_parametara)] LOOP  
  statement1;  
  statement2;  
  . . .  
END LOOP;
```

Kursorska FOR petlja

- Obavezna deklaracija kursorskog područja
- Automatsko otvaranje, preuzimanje torki i zatvaranje kursora
- Slogovsku promenljivu `record_var` nije potrebno eksplicitno deklarirati

Primer eksplicitno deklarisanog kursora s parametrima i upotrebe kursorske FOR petlje.

```
DECLARE
```

```
    Ukup_Plt NUMBER;
```

```
    CURSOR spisak_rad (D_gran radnik.Mbr%TYPE, G_gran  
radnik.Mbr%TYPE)
```

```
IS
```

```
SELECT *
```

```
FROM radnik
```

```
WHERE Mbr BETWEEN D_gran AND G_gran;
```

```
BEGIN
```

```
    Ukup_Plt := 0;
```

```
    FOR p_tek_red IN spisak_rad (01, 99) LOOP
```

```
        Ukup_Plt := Ukup_Plt + p_tek_red.Plt;
```

```
    END LOOP;
```

```
        DBMS_OUTPUT.PUT_LINE('Plata je: ' || Ukup_Plt);
```

```
END;
```

Kursorska FOR petlja sa implicitnom deklaracijom kursora

```
FOR record_var IN (SELECT ...) LOOP  
    statement1;  
    statement2;  
    . . .  
END LOOP;
```

Kursorska FOR petlja sa implicitnom deklaracijom kursora

- Kursorsko područje se ne deklariše eksplicitno
- Automatsko otvaranje, preuzimanje torki i zatvaranje kursora
- Slogovsku promenljivu `record_var` nije potrebno eksplicitno deklarirati

Primer upotrebe kursorske FOR petlje, s implicitno deklaranim kursorom

```
DECLARE
```

```
    Ukup_Plt NUMBER;
```

```
BEGIN
```

```
    Ukup_Plt := 0;
```

```
    FOR p_tek_red IN (SELECT * FROM radnik  
                      WHERE Mbr BETWEEN 01 AND 99)
```

```
    LOOP      -- otvoren kursor, izvršava se SELECT
```

```
        Ukup_Plt := Ukup_Plt + p_tek_red.Plt;
```

```
    END LOOP;      -- zatvoren kursor
```

```
        DBMS_OUTPUT.PUT_LINE('Plata je: ' || Ukup_Plt);
```

```
END;
```

Zadaci

- Napisati PL/SQL blok koji će preuzeti sve torke iz tabele Projekat i prebaciti ih u PL/SQL tabelarnu kolekciju. Zatim će, redom, odštampati sve elemente tako dobijene tabelarne kolekcije.

Rešenje prvog zadatka

```
DECLARE
TYPE T_Projekat IS TABLE OF
  Projekat%ROWTYPE INDEX BY BINARY_INTEGER;
Tabela T_Projekat;
i BINARY_INTEGER:=0;
BEGIN
FOR rec IN (SELECT * FROM Projekat) LOOP
  Tabela(i):=rec;
  i:=i+1;
END LOOP;
i:= Tabela.FIRST;
WHILE i<=Tabela.LAST LOOP
  DBMS_OUTPUT.PUT_LINE('Naziv projekta: ' || Tabela(i).nap);
  DBMS_OUTPUT.PUT_LINE('Sifra rukovodioca: ' || Tabela(i).ruk);
  DBMS_OUTPUT.PUT_LINE('Narucilac projekta: ' || Tabela(i).nar);
  i:=Tabela.NEXT(i);
END LOOP;
END;
```

Zadaci

- Napisati PL/SQL blok koji će preuzeti sve torke iz tabele Projekat, uređene u opadajućem redosledu šifri projekata, i prebaciti ih u PL/SQL tabelarnu kolekciju. Uz svaku preuzetu torku iz tabele Projekat, treba inicijalizovati novu kolekciju koja će sadržati skup svih matičnih brojeva radnika, koji su angažovani na datom projektu. Zatim treba, redom, odštampati sve torke iz kolekcije projekata, a uz svaku torku iz kolekcije projekata treba prikazati matične brojeve svih radnika koji su angažovani na tom projektu.

Rešenje drugog zadatka

```
DECLARE
TYPE T_Projekat IS TABLE OF
  Projekat%ROWTYPE INDEX BY BINARY_INTEGER;
TYPE T_SifRad IS TABLE OF
  Radnik.mbr%TYPE INDEX BY BINARY_INTEGER;
TYPE T_Radnici IS TABLE OF
  T_SifRad INDEX BY BINARY_INTEGER;
TabelaP T_Projekat;
TabelaR T_Radnici;
i BINARY_INTEGER:=0;
j BINARY_INTEGER:=1;
BEGIN
FOR rec IN (SELECT * FROM Projekat ORDER BY spr DESC) LOOP
  TabelaP(i):=rec;
  FOR rec1 IN (SELECT mbr FROM RadProj WHERE spr = rec.spr) LOOP
    TabelaR(i)(j):=rec1.mbr;
    j:=j+1;
  END LOOP;
  i:=i+1;
END LOOP;
i:= TabelaP.FIRST;
WHILE i<=TabelaP.LAST LOOP
  DBMS_OUTPUT.PUT_LINE('Naziv projekta: ' || TabelaP(i).nap);
  DBMS_OUTPUT.PUT_LINE('Sifra rukovodioca: ' || TabelaP(i).ruk);
  DBMS_OUTPUT.PUT_LINE('Narucilac projekta: ' || TabelaP(i).nar);
  j:= TabelaR(i).FIRST;
  WHILE j<=TabelaR(i).LAST LOOP
    DBMS_OUTPUT.PUT_LINE('Maticni broj radnika: ' || TabelaR(i)(j));
    j:= TabelaR(i).NEXT(j);
  END LOOP;
  i:=TabelaP.NEXT(i);
END LOOP;
END;
```

Rešenje drugog zadatka

```
DECLARE
TYPE T_SifRad IS TABLE OF
Radnik.mbr%TYPE INDEX BY BINARY_INTEGER;
TYPE T_Projekat IS RECORD (
ProjPodaci Projekat%ROWTYPE,
Radnici T_SifRad);
TYPE T_Projekti IS TABLE OF
T_Projekat INDEX BY BINARY_INTEGER;

TabelaP T_Projekti;
i BINARY_INTEGER:=0;
j BINARY_INTEGER:=1;

BEGIN
FOR rec IN (SELECT * FROM Projekat ORDER BY spr DESC) LOOP
TabelaP(i).ProjPodaci:=rec;
FOR rec1 IN (SELECT mbr FROM RadProj WHERE spr = rec.spr) LOOP
TabelaP(i).Radnici(j):=rec1.mbr;
j:=j+1;
END LOOP;
i:=i+1;
END LOOP;
i:= TabelaP.FIRST;
WHILE i<=TabelaP.LAST LOOP
DBMS_OUTPUT.PUT_LINE('Naziv projekta: ' || TabelaP(i).ProjPodaci.nap);
DBMS_OUTPUT.PUT_LINE('Sifra rukovodioca: ' || TabelaP(i).ProjPodaci.ruk);
DBMS_OUTPUT.PUT_LINE('Narucilac projekta: ' || TabelaP(i).ProjPodaci.nar);
DBMS_OUTPUT.PUT_LINE("");
IF TabelaP(i).Radnici.COUNT != 0 THEN
DBMS_OUTPUT.PUT_LINE('Radnici: ');
DBMS_OUTPUT.PUT_LINE("");
j:= TabelaP(i).Radnici.FIRST;
WHILE j<=TabelaP(i).Radnici.LAST LOOP
DBMS_OUTPUT.PUT_LINE('Maticni broj radnika: ' || TabelaP(i).Radnici(j));
j:= TabelaP(i).Radnici.NEXT(j);
END LOOP;
ELSE
DBMS_OUTPUT.PUT_LINE('Nijedan radnik ne radi na projektu "' || TabelaP(i).ProjPodaci.nap || '".');
END IF;
i:=TabelaP.NEXT(i);
DBMS_OUTPUT.PUT_LINE("");
END LOOP;
END;
```

Zadatak

- Napisati PL/SQL blok koji će prikazati koliko radnika nema ni najmanju ni najveću platu.

Rešenje

```
declare
plamin radnik.plt%type;
plamax radnik.plt%type;
ukupno NUMBER;

begin
ukupno:=0;
select min(plt) into plamin from radnik;
select max(plt) into plamax from radnik;
DBMS_OUTPUT.PUT_LINE('Min plata: ' || plamin);
DBMS_OUTPUT.PUT_LINE('Max plata: ' || plamax);
for rec_radnik in (select * from radnik) loop
    if (rec.plt > plamin and rec.plt < plamax) then
        ukupno:=ukupno+1;
    end if;
end loop;
DBMS_OUTPUT.PUT_LINE('Broj radnika: ' || ukupno);
end;
```

Obrada izuzetaka

- Definicija izuzetka
 - Događaj koji izaziva prekid normalnog toka izvođenja programa
 - Zahteva pisanje posebnog programskog koda, koji će biti izvršen u slučaju nastupanja izuzetka – "obrada izuzetka"
 - **EXCEPTION** – programska celina za obradu PL/SQL izuzetaka
 - **Exception Handler** - deo PL/SQL bloka koji obrađuje izuzetke
 - Prelaskom na exception handler, nemoguće je vratiti tok izvođenja programa nazad, u BEGIN deo programskog bloka

Vrste i deklarisanje izuzetaka u PL/SQL-u

- **Tipovi PL/SQL izuzetaka**

- Predefinisani
- Nepredefinisani
- Korisnički definisan

- **Predefinisani izuzetak**

- Unapred definisano ime
- Unapred povezan sa ORA (DBMS) greškom koja ga izaziva
- Programer može i direktno izazvati ovu vrstu izuzetka, ali je prirodno da se on izaziva automatski, pojavom greške s kojom je povezan

Lista predefinisanih izuzetaka

NAZIV IZUZETKA	ORACLE KOD GREŠKE	Vrednost SQLCODE
ACCESS_INTO_NULL	ORA-06530	-6530
CASE_NOT_FOUND	ORA-06592	-6592
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-0001
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+0100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
STORAGE_ERROR	ORA-06500	-6500
SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
TIMEOUT_ON_RESOURCE	ORA-00051	-0051
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

Vrste i deklarisanje izuzetaka u PL/SQL-u

- **Nepredefinisani izuzetak**
 - Nema unapred definisano ime
 - Nije unapred povezan sa ORA (DBMS) greškom koja ga izaziva
 - Programer ga mora, eksplicitno, deklarirati i povezati sa ORA (DBMS) greškom koja će ga izazivati
 - Programer može i direktno izazvati ovu vrstu izuzetka, ali je prirodno da se on izaziva automatski, pojavom greške s kojom je povezan
 - Deklarisanje i povezivanje nepredefinisanog izuzetka – u deklarativnom delu programa

Vrste i deklarisanje izuzetaka u PL/SQL-u

- **Nepredefinisani izuzetak**

naziv_izuzetka EXCEPTION;

**PRAGMA EXCEPTION_INIT(naziv_izuzetka,
kod_ORA_greške);**

Primeri deklarisanja i inicijalizovanja nepredefinisanog izuzetka

```
DECLARE
  Delete_RefInt_ERR EXCEPTION;
  PRAGMA EXCEPTION_INIT (Delete_RefInt_ERR, -2292);
BEGIN
  ...
EXCEPTION
  ...
END;
```

```
DECLARE
  InsUpd_RefInt_ERR EXCEPTION;
  PRAGMA EXCEPTION_INIT (InsUpd_RefInt_ERR, -2291);
BEGIN
  ...
EXCEPTION
  ...
END;
```

Vrste i deklarisanje izuzetaka u PL/SQL-u

- **Korisnički definisani izuzetak**
 - Nema unapred definisano ime
 - Ne povezuje se sa ORA (DBMS) greškom koja bi ga izazivala
 - Programer isključivo direktno izazva ovu vrstu izuzetka, posebnom naredbom
 - Deklarisanje korisničkog izuzetka – u deklarativnom delu programa

`naziv_izuzetka EXCEPTION;`

Vrste i deklarisanje izuzetaka u PL/SQL-u

- **Korisnički definisani izuzetak**
 - Izazivanje korisničkog izuzetka – u proceduralnom delu programa, ili u delu programa za obradu izuzetaka – naredba RAISE

RAISE [naziv_izuzetka];

Primer deklarisanja i izazivanja korisnički definisanog izuzetka

DECLARE

Izuzetak EXCEPTION;

BEGIN

...

RAISE Izuzetak;

...

EXCEPTION

...

END;

Obrada izuzetaka u EXCEPTION delu programa

EXCEPTION

WHEN exception1 [OR exception2 . . .] THEN

statement1;

statement2;

. . .

[WHEN exception3 [OR exception4 . . .] THEN

statement1;

statement2;

. . .

]

[WHEN OTHERS THEN

statement1;

statement2;

. . .

]

Obrada izuzetaka u EXCEPTION delu programa

- Klauzula OTHERS
 - pokriva sve ostale izuzetke, koji u EXCEPTION bloku nisu prethodno eksplicitno navedeni
 - ako se navodi, OTHERS je uvek poslednji obrađeni izuzetak u EXCEPTION bloku

Pravila upravljanja tokom izvođenja programa

- U slučaju izazivanja izuzetka prekida se normalni tok izvođenja programa i programski tok se preusmerava u Exception Handler – EXCEPTION deo PL/SQL bloka
- Traži se prva WHEN klauzula, koja sadrži naziv izuzetka koji je nastao, ili sadrži naziv OTHERS

Pravila upravljanja tokom izvođenja programa

- **OBRAĐENI IZUZETAK**

- izuzetak za koji postoji odgovarajuća WHEN klauzula u EXCEPTION delu

- **NEOBRAĐENI IZUZETAK**

- izuzetak za koji ne postoji odgovarajuća WHEN klauzula u EXCEPTION delu, niti postoji klauzula WHEN OTHERS

- U slučaju obrađenog izuzetka, izvršava se imperativni blok odgovarajuće WHEN klauzule i završava se izvođenje PL/SQL bloka

Pravila upravljanja tokom izvođenja programa

- izvršavanje bloka koji obrađuje izuzetak može biti uspešno, ili neuspešno
 - **USPEŠNO:** ako nije došlo do pojave istog, ili nekog drugog izuzetka (greške) – tok upravljanja programom vraća se u nadređeni kontekst, na mesto odakle je PL/SQL blok pozvan
 - **NEUSPEŠNO:** ako je došlo do pojave istog, ili nekog drugog izuzetka (greške) – tok upravljanja programom vraća se u nadređeni kontekst, a izuzetak se prosleđuje u nadređeni kontekst – videti tekst koji sledi

Pravila upravljanja tokom izvođenja programa

- U slučaju neobrađenog izuzetka, izuzetak se prosleđuje u pozivajući kontekst:
 - u nadređeni blok, iz kojeg je dati blok pozvan
 - izaziva se isti izuzetak u tom bloku i prenosi se tok upravljanja programom na njegov EXCEPTION deo – ova situacija može rekurzivno da se ponavlja sve do pozivajućeg radnog okruženja
 - u pozivajuće radno okruženje (npr. SQL*Plus)
 - izuzetak se, u radnom okruženju, ispoljava kao neobrađena greška
- U slučaju pojave greške u radnom okruženju, poništavaju se samo efekti izvođenja celokupnog PL/SQL bloka, ali se transakcija niti poništava, niti potvrđuje

Primer obrade korisnički definisanih izuzetaka

- Koju vrednost će imati *r*, za zadatu vrednost *x*: 0, 1, 2 ili 5?

```
ACCEPT X PROMPT 'Unesite vrednost za x'
```

```
DECLARE
```

```
  A EXCEPTION;
```

```
  B EXCEPTION;
```

```
  C EXCEPTION;
```

```
  D EXCEPTION;
```

```
  r NUMBER;
```

```
BEGIN
```

```
  BEGIN
```

```
    IF &x = 0 THEN RAISE A;
```

```
    ELSIF &x = 1 THEN RAISE B;
```

```
    ELSIF &x = 2 THEN RAISE C;
```

```
    ELSE RAISE D;
```

```
    END IF;
```

```
  EXCEPTION
```

```
    WHEN A THEN r := 1;
```

```
    DBMS_OUTPUT.PUT_LINE ('Za izuzetak A r je ' || r);
```

```
  END;
```

```
  r := 2;
```

```
  DBMS_OUTPUT.PUT_LINE ('Nema izuzetka i r je ' || r);
```

```
EXCEPTION
```

```
  WHEN B THEN
```

```
    r := 3;
```

```
    DBMS_OUTPUT.PUT_LINE ('Za izuzetak B r je ' || r);
```

```
  WHEN OTHERS THEN
```

```
    r:=10;
```

```
    DBMS_OUTPUT.PUT_LINE ('Za sve navedene izuzetke (C,D) r je ' || r);
```

```
END;
```

Primer obrade nepredefinisanog, deklarisanog izuzetka

```
DECLARE
  Delete_RefInt_ERR EXCEPTION;
  PRAGMA EXCEPTION_INIT (Delete_RefInt_ERR, -2292);
BEGIN
  DELETE FROM Projekat
  WHERE Spr = &p_spr;
  COMMIT;
EXCEPTION
  WHEN Delete_RefInt_ERR THEN
    DBMS_OUTPUT.PUT_LINE ('Nije dozvoljeno
    brisanje projekta ' || TO_CHAR(&p_spr) || '. Postoje
    povezani radnici.');
```

```
ROLLBACK;
END;
```

Primer obrade predefinisiranog izuzetka, vezanog za SELECT naredbu

```
ACCEPT P_ruk PROMPT 'Unesi sifru rukovodioca'
DECLARE
    V_Proj Projekat%ROWTYPE;
BEGIN
    SELECT *
    INTO   V_Proj
    FROM   Projekat
    WHERE  Ruk = &P_ruk;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nije selektovana ni jedna toraka
...');
        RAISE NO_DATA_FOUND;
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Selektovano je vise od jedne
torke ...');
        RAISE;
END;
```


Upotreba UPDATE i DELETE naredbe u kombinaciji s kursorima

SELECT ...

FROM ...

FOR UPDATE [OF *column_list*][NOWAIT];

UPDATE ... | DELETE ...

WHERE CURRENT OF naziv_kursora

Primer

- Primer eksplicitno deklarisanog kursora s upotrebom naredbe UPDATE...WHERE CURRENT OF, slogovskom promenljivom i eksplicitnog izazivanja predefinisiranog izuzetka NO_DATA_FOUND.

Primer

```
DECLARE
  L_tek_red radnik%ROWTYPE;
  CURSOR spisak_rad IS          -- eksplicitno deklarisan kursor
  SELECT *
  FROM radnik
  WHERE Mbr BETWEEN 01 AND 99
  FOR UPDATE OF Prz NOWAIT;
  Ukup_plt NUMBER;
BEGIN
  Ukup_Plt := 0;
  LOOP
    IF NOT spisak_rad%ISOPEN THEN      -- Da li je kursor otvoren?
      OPEN spisak_rad;                -- otvaranje kursora, izvršava se SELECT
    END IF;
    FETCH spisak_rad INTO L_tek_red;
    EXIT WHEN spisak_rad%NOTFOUND;    -- uslov izlaska iz petlje
    Ukup_Plt := Ukup_Plt + L_tek_red.Plt;
    UPDATE radnik
    SET Prz = Prz || ' x'
    WHERE CURRENT OF spisak_rad;      -- modifikacija tekuće torke u BP
    IF SQL%NOTFOUND THEN              -- Da li torka nije modifikovana?
      RAISE NO_DATA_FOUND;           -- izazivanje izuzetka
    END IF;
  END LOOP;
  CLOSE spisak_rad;                  -- zatvaranje kursora
  COMMIT;                             -- potvrđivanje transakcije
  DBMS_OUTPUT.PUT_LINE('Ukupna plata: '||Ukup_Plt);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Izuzetak!');
    ROLLBACK;                         -- poništavanje transakcije
    RAISE;                             -- ponovno izazivanje istog izuzetka
END;
```

Funkcije za obradu grešaka u EXCEPTION delu PL/SQL bloka

- SQLCODE

Rezultat funkcije je numerička vrednost. SQLCODE vraća broj greške, koji odgovara poslednjem izazvanom izuzetku. Izvan EXCEPTION bloka, vrednost funkcije je **0**.

- SQLERRM [(*<broj_greške>*)]

Rezultat funkcije je karakter vrednost. SQLERRM vraća broj i opis greške, koji odgovara poslednjem izazvanom izuzetku. Izvan EXCEPTION bloka, vrednost funkcije je **'ORA-0000: normal, successful completion'**.

Primer obrade predefinisiranog izuzetka, vezanog za SELECT naredbu, uz upotrebu funkcija za obradu grešaka

```
ACCEPT P_ruk PROMPT 'Unesi sifru rukovodioca'
DECLARE
    V_Proj Projekat%ROWTYPE;
BEGIN
    SELECT *
    INTO V_Proj
    FROM Projekat
    WHERE Ruk = &P_ruk;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nije selektovana ni jedna torka ...');
        DBMS_OUTPUT.PUT_LINE(SQLCODE || ' ' || SQLERRM);
        RAISE NO_DATA_FOUND;
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Selektovano je vise od jedne torke ...');
        DBMS_OUTPUT.PUT_LINE(SQLCODE || ' ' || SQLERRM);
        RAISE TOO_MANY_ROWS;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLCODE || ' ' || SQLERRM);
        RAISE;
END;
```

Naredba za izazivanje korisničke ORA greške

- Način da se programskim putem izazove ORA (DBMS) greška

```
raise_application_error (error_number, message [,  
    {TRUE | FALSE});
```

- Broj greške (error_number) može biti zadat u intervalu od -20000 do -20999.

Naredba za izazivanje korisničke ORA greške

- **TRUE** – kod i poruka o grešci se memorišu na stek grešaka
 - omogućen prikaz svih grešaka na pozivajućem putu PL/SQL blokova – pogodno za veliku dubinu pozivanja
- **FALSE** – kod i poruka o grešci brišu prethodni sadržaj steka grešaka
 - prikazuje se samo poslednji kod i poruka o grešci – jednostavniji pristup

Primer

- Primer obrade predefinisiranog izuzetka, vezanog za SELECT naredbu, uz upotrebu funkcija za obradu grešaka i naredbe za izazivanje korisničke ORA greške

Rešenje

```
ACCEPT P_ruk PROMPT 'Unesi sifru rukovodioca'
DECLARE
    V_Proj Projekat%ROWTYPE;
BEGIN
    SELECT *
    INTO V_Proj
    FROM Projekat
    WHERE Ruk = &P_ruk;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nije selektovana ni jedna torka ...');
        DBMS_OUTPUT.PUT_LINE(SQLCODE || ' ' || SQLERRM);
        RAISE NO_DATA_FOUND;
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Selektovano je vise od jedne torke ...');
        DBMS_OUTPUT.PUT_LINE(SQLCODE || ' ' || SQLERRM);
        RAISE TOO_MANY_ROWS;
    WHEN OTHERS THEN
        Raise_Application_Error(-20000, SQLCODE || ' ' || SQLERRM);
END;
```