Sistemi baza podataka

Slavica Aleksić

slavica@uns.ns.ac.yu

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 predefinisanog 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 torka
    ...');
            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 predefinisanog 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 predefinisanog 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 predefinisanog 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;
```

Procedure u PL/SQL-u

- Potprogram
  - Imenovani PL/SQL blok
  - Osposobljen da primi ulazne vrednosti i preda izlazne rezultate u pozivajuće okruženje
  - Egzistira
    - u rečniku podataka DBMS-a, ili
    - unutar klijentskog programa (razvijenog putem nekog od alata Oracle Developer Suite-a)

Procedure u PL/SQL-u

- Vrste
  - procedura
    - predstavlja naredbu koja se poziva kao i bilo koja druga naredba – navođenjem naziva
  - funkcija
    - predstavlja unarni operator koji se koristi u izrazima i namenjen je da vrati izračunatu vrednost u izraz iz kojeg je pozvan

Vrste PL/SQL blokova

- Neimenovani (anonimni) blok

```
[DECLARE
    ...                    -- Deklarativni deo bloka
]
BEGIN
    ...                    -- Izvršni deo bloka
[EXCEPTION
    ...                    -- Deo bloka za obradu izuzetaka
]
END;
```

Imenovani (programski) blok – procedura ili funkcija

```
Zaglavlje_programskog_bloka
IS | AS
[   ...                    -- Deklarativni deo bloka
]
BEGIN
    ...                    -- Izvršni deo bloka
[EXCEPTION
    ...                    -- Deo bloka za obradu izuzetaka
]
END;
```

Vrste procedura i funkcija (imenovanih programskih blokova)

- Serverska procedura ili funkcija
  - procedura ili funkcija, kreirana na nivou DBMS i memorisana u rečniku podataka DBMS
  - egzistira u rečniku podataka u dva oblika:
    - izvornom (source kod)
    - prekompajliranom (P-kod – izvršni kod, interpretabilan od strane DBMS i PL/SQL Engine-a)
- Lokalna procedura ili funkcija
  - procedura ili funkcija, deklarisana unutar nekog PL/SQL bloka (programa)
- Klijentska procedura ili funkcija
  - procedura ili funkcija, deklarisana u okviru nekog alata iz Oracle Developer Suite
  - nalazi se i izvršava na srednjem sloju (aplikativnom serveru)

Naredba za kreiranje procedura

```
CREATE [OR REPLACE] PROCEDURE [schema.]procedure_name
    [(parameter1 [IN | OUT | IN OUT] datatype1 [DEFAULT def_value],
      parameter2 [IN | OUT | IN OUT] datatype2 [DEFAULT def_value],
            . . .
     )]
IS | AS
[    ...                    -- Deklarativni deo bloka
]
BEGIN
    ...                     -- Izvršni deo bloka
[EXCEPTION
    ...                     -- Deo bloka za obradu izuzetaka
]
END[procedure_name];
```

Naredba za kreiranje procedura

- IN -- specifikacija ulaznog parametra procedure
  - vrednost parametra se zadaje pri pozivu procedure i ne sme da se menja unutar procedure
  - dozvoljeno je zadavanje DEFAULT vrednosti parametra
  - prenos parametra po referenci

Naredba za kreiranje procedura

- OUT -- specifikacija izlaznog parametra procedure
  - procedura generiše i vraća vrednost parametra u pozivajuće okruženje
  - nije dozvoljeno zadavanje DEFAULT vrednosti parametra
  - prenos parametra po vrednosti
  - Druga varijanta: OUT NOCOPY
    - specifikacija izlaznog parametra s prenosom po referenci

Naredba za kreiranje procedura

- IN OUT - specifikacija ulazno-izlaznog parametra procedure
  - vrednost parametra se zadaje pri pozivu procedure, može da se menja unutar procedure i vraća se izmenjena vrednost u pozivajuće okruženje
  - nije dozvoljeno je zadavanje DEFAULT vrednosti parametra
  - prenos parametra po vrednosti
  - Druga varijanta: IN OUT NOCOPY
    - specifikacija ulazno-izlaznog parametra s prenosom po referenci

Naredbe za menjanje i brisanje serverskih procedura

ALTER PROCEDURE
[schema.]procedure_name COMPILE;

DROP PROCEDURE
[schema.]procedure_name;

Primer kreiranja procedure s deklaracijom ulaznih parametara

```
CREATE OR REPLACE PROCEDURE P_INS_Radnik
      (P_Mbr IN Radnik.Mbr%TYPE,
       P_Prz IN Radnik.Prz%TYPE,
       P_Ime IN Radnik.Ime%TYPE,
       P_Plt IN Radnik.Plt%TYPE,
       P_God IN Radnik.God%TYPE DEFAULT SYSDATE,
       P_Pre IN Radnik.Pre%TYPE DEFAULT NULL
      )
IS
BEGIN
      INSERT INTO radnik (Mbr, Prz, Ime, Plt, God, Pre)
      VALUES (P_Mbr, P_Prz, P_Ime, P_Plt, P_God, P_Pre);
      COMMIT;
EXCEPTION
      WHEN DUP_VAL_ON_INDEX THEN
            ROLLBACK;
            Raise_application_error (-20000, 'Dupla vrednost kljuca.');
      WHEN VALUE_ERROR THEN
            ROLLBACK;
            Raise_application_error (-20000, 'Greska u vrednosti podatka.');
END P_INS_Radnik;
```

Primer kreiranja procedure s deklaracijom ulaznih i izlaznih parametara

```
CREATE OR REPLACE PROCEDURE P_SEL_Projekat
   (P_Spr  IN  Projekat.Spr%TYPE,
    P_Proj OUT Projekat%ROWTYPE)
IS
BEGIN
   SELECT *
   INTO   P_Proj
   FROM   Projekat
   WHERE Spr = P_Spr;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
         P_Proj := NULL;
END P_SEL_Projekat;
```

Primer kreiranja procedure s deklaracijom ulaznog parametra tipa tabele

```
CREATE OR REPLACE TYPE T_ProjS AS OBJECT (Spr NUMBER(3), Ruk NUMBER(3), Nap
    VARCHAR2(30), Nar VARCHAR2(30));

CREATE OR REPLACE TYPE T_Proj_ROWS AS TABLE OF T_ProjS;

CREATE OR REPLACE PROCEDURE P_INS_Projekat_ROWS
    (P_Rows IN T_Proj_ROWS)
IS
    i NUMBER;
BEGIN
    i := P_Rows.FIRST;
    WHILE i IS NOT NULL LOOP
            INSERT INTO Projekat (Spr, Ruk, Nap, Nar)
            VALUES    (P_Rows(i).Spr, P_Rows(i).Ruk, P_Rows(i).Nap, P_Rows(i).Nar);
            i := P_Rows.NEXT(i);
    END LOOP;
    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
            ROLLBACK;
            Raise_application_error (-20000, 'Dupla vrednost kljuca.');
    WHEN VALUE_ERROR THEN
            ROLLBACK;
            Raise_application_error (-20000, 'Greska u vrednosti podatka.');
END P_INS_Projekat_ROWS;
```

Pozivanje PL/SQL procedure iz drugog PL/SQL bloka

- Pozivanje procedure

Naziv_procedure [([formalni_param1 =>] stvarni_param1,
[([formalni_param2 =>] stvarni_param2,...)]

- Umesto IN formalnih parametara, kao stvarni parametri, mogu se pojaviti:
  - izrazi odgovarajućeg tipa, ili
  - promenljive odgovarajućeg tipa
- Umesto IN OUT i OUT formalnih parametara, kao stvarni parametri, mogu se pojaviti samo promenljive odgovarajućeg tipa.

Primeri mogućih načina pozivanja prethodno kreiranih procedura

```
P_INS_Radnik(200, 'Antic', 'Ante', 20000,
   TO_DATE('01.10.1965', 'DD.MM.YYYY'), 2000);
P_INS_Radnik(210, 'Anic', 'Ana', 22000,
   TO_DATE('01.10.1975', 'DD.MM.YYYY'));
P_INS_Radnik(220, 'Djurisic', 'Djuro', 24000);
P_INS_Radnik(P_Ime => 'Danko', P_Prz =>
   'Danic', P_Mbr => 230, P_Plt => 21000);
P_INS_Radnik(230, P_Ime => 'Danko', P_Prz =>
   'Danic', P_Plt => 18000, P_Pre => NULL);
```

Primeri mogućih načina pozivanja prethodno kreiranih procedura

```
DECLARE
   V_Proj Projekat%ROWTYPE;
BEGIN
   P_SEL_Projekat(&P_Spr, V_Proj);
   DBMS_OUTPUT.PUT_LINE('Projekat: ' || V_Proj.Spr || ' '
   || V_Proj.Nap);
END;
DECLARE
   V_Proj Projekat%ROWTYPE;
BEGIN
   P_SEL_Projekat(P_Proj => V_Proj, P_Spr => &P_Spr);
   DBMS_OUTPUT.PUT_LINE('Projekat: ' || V_Proj.Spr || ' '
   || V_Proj.Nap);
END;
```

Primeri mogućih načina pozivanja prethodno kreiranih procedura

```
DECLARE
   ProjR T_Proj_ROWS := T_Proj_ROWS
   (T_ProjS(80, 10, 'Kurs I', 'RAF'), T_ProjS(90, 10,
   'Kurs II', 'RAF'));
BEGIN
   P_INS_Projekat_ROWS(ProjR);
   P_INS_Projekat_ROWS(T_Proj_ROWS());
   P_INS_Projekat_ROWS(T_Proj_ROWS
   (T_ProjS(100, 10, 'Kurs III', 'RAF')));
END;
```

Deklarisanje i pozivi lokalnih procedura

- Deklarisanje se obavlja unutar deklarativnog dela PL/SQL bloka (programa)
- Pozivanje – na uobičajen način

Deklarisanje i pozivi lokalnih procedura

```
PROCEDURE procedure_name
    [(parameter1 [IN | OUT | IN OUT] datatype1 [DEFAULT def_value],
      parameter2 [IN | OUT | IN OUT] datatype2 [DEFAULT def_value],
            . . .
     )
    ]
IS | AS
[    ...                    -- Deklarativni deo bloka
]
BEGIN
    ...                     -- Izvršni deo bloka
[EXCEPTION
    ...                     -- Deo bloka za obradu izuzetaka
]
END;
```

Primer deklaracije i poziva lokalne funkcije

```
CREATE OR REPLACE PROCEDURE A
IS
   PROCEDURE B (P_1 IN NUMBER)
   IS
   BEGIN
        NULL;
   END;
BEGIN
   B(10);
END;
```

Primeri

- Napisati lokalnu proceduru koja će, u okviru jedne transakcije, putem kursora, preuzimati, redom, sve torke iz tabele Projekat i prebacivati ih, jednu po jednu, u PL/SQL tabelarnu kolekciju. Takva tabelarna kolekcija treba da predstavlja izlazni parametar procedure.
- Proveriti ispravnost rada procedure pozivima na konkretnim primerima

Rešenje

```
declare
TYPE T_Proj_zad1 IS TABLE OF projekat%rowtype INDEX BY BINARY_INTEGER;
Tabela T_Proj_zad1;
i integer;
PROCEDURE P_Projekat_Zad1(Tabela OUT NOCOPY T_Proj_zad1)
IS
      i BINARY_INTEGER:=0;
BEGIN
FOR rec IN (SELECT * FROM Projekat) LOOP
   Tabela(i):=rec;
   i:=i+1;
END LOOP;
END;
begin
P_Projekat_Zad1(Tabela);
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;
```

Primeri

- Napisati proceduru koja će, u okviru jedne transakcije, putem kursora, preuzimati, redom, sve torke iz tabele Projekat, uređene u opadajućem redosledu šifri projekata, i prebacivati ih u PL/SQL tabelarnu kolekciju. Uz svaku preuzetu torku iz tabele Projekat, treba u okviru elementa te kolekcije, inicijalizovati novu kolekciju koja će sadržati skup svih matičnih brojeva radnika, koji su angažovani na datom projektu. Skup matičnih brojeva radnika, za dati projekat, treba selektovati putem posebnog kursora. Tabelarna kolekcija selektovanih projekata treba da predstavlja izlazni parametar procedure.
- Proveriti ispravnost rada procedure pozivima na konkretnim primerima.

Primeri

- Napisati proceduru koja će, u okviru jedne transakcije, putem kursora, selektovati sve radnike, uređene (sortirane) po zadatom kriterijumu. Kriterijum uređivanja (lista u ORDER BY klauzuli) sastoji se uvek od 3 elementa, čije vrednosti treba preuzeti putem parametara procedure. Obezbediti programsko (dinamičko) formiranje SELECT naredbe kursorskog područja. Selektovane torke treba preneti u tabelarnu promenljivu (kolekciju). Tabelarna kolekcija selektovanih radnika treba da predstavlja izlazni parametar procedure.

  Neki primeri mogućih SELECT naredbi kursora:
  SELECT * FROM RADNIK ORDER BY 3, 2, 1
  SELECT * FROM RADNIK ORDER BY Prz, Ime, Mbr
  SELECT * FROM RADNIK ORDER BY Mbr, Prz, Ime

- Proveriti ispravnost rada procedure pozivima na konkretnim primerima.