acs
applied computer science

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

DAIS
Data Science and
Information Systems Group

Serversko programiranje

Baze podataka

Uloga jezika PL/SQL i struktura PL/SQL programa

- PL/SQL - jezik III generacije
- PL/SQL - predstavlja proceduralno proširenje SQL-a
- PL/SQL se može koristiti iz različitih okruženja
  - SQL*Plus
  - Oracle Developer Suite (Forms, Reports, Oracle Portal, Oracle Discoverer)
  - SQL Developer

Osobine jezika PL/SQL

- Strukturirano programiranje i organizacija programa po blokovima
- Proceduralna podrška osnovnih struktura: sekvenca, selekcija i iteracija
- Podrška neproceduralnog jezika SQL
- Mogućnost deklarisanja promenljivih i konstanti i upotreba osnovnih i složenih tipova podataka
- Upotreba kursora - proceduralna obrada rezultata SQL SELECT naredbe
- Mogućnost obrade grešaka i izuzetaka, indikovanih od strane DBMS ORACLE

Primer trigera

- Primer trigera koji kontroliše da li korisnik pokušava za vrednost kolone Pre u tabeli Radnik da zada negativnu vrednost. Ukoliko je to slučaj, umesto vrednosti koju zadaje korisnik, kolona Pre treba da dobije vrednost 0.

```
CREATE OR REPLACE TRIGGER Trg_Radnik_Pre_INSUPD
BEFORE INSERT OR UPDATE OF Pre
ON RADNIK
FOR EACH ROW
WHEN (NEW.Pre < 0)
BEGIN
    :NEW.Pre := 0;
END Trg_Radnik_Pre_INSUPD;
```

Osnovna struktura PL/SQL bloka

- **Tipovi PL/SQL blokova**
  - anonimni (netipizovani)
  - tipizovani (procedura, funkcija)

Vrste PL/SQL blokova

- Neimenovani (anonimni) blok

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

Struktura anonimnog PL/SQL bloka

[DECLARE

Deklarativni (neobavezni) deo programa:

* deklaracija i inicijalizacija promenljivih
* deklaracija i inicijalizacija konstanti
* deklaracija tipova podataka
* deklaracija kursora
* deklaracija izuzetaka
* deklaracija procedura i funkcija

]

BEGIN

Izvršni (obavezni) deo programa:

* Proceduralne naredbe
* SQL naredbe

[EXCEPTION

Deo za obradu izuzetaka (neobavezni):

* WHEN <izuzetak> THEN <blok izvršnih naredbi>

]

END;

Primer jednog PL/SQL bloka

```
-- Ovo je oznaka za jednolinijski komentar
/* Ovo je način za definisanje višelinijskog komentara */
DECLARE                                 -- Deklarativni deo bloka
        Br_torki NUMBER(6) := 0;                -- Deklarisana i inicijalizovana lokalna promenljiva
        L_OznDeo Deo.OznDeo%TYPE;   -- Deklaracija saglasno tipu kolone iz tabele Deo
BEGIN                                   -- Izvršni deo bloka
        SELECT COUNT(*)
        INTO Br_torki
        FROM Deo_koji_se_dobavlja
        WHERE OznDeo = :p_OznDeo;    -- Referenca na promenljivu iz pozivajućeg okruženja
        IF Br_torki = 0 THEN
            SELECT COUNT(*)
            INTO Br_torki
            FROM Deo_iz_proizvodnje
            WHERE OznDeo = :p_OznDeo;           -- Referenca na promenljivu iz pozivajućeg okruženja
            IF Br_torki = 0 THEN
                        RAISE NO_DATA_FOUND;
            END IF;
        END IF;
EXCEPTION                               -- Deo za obradu izuzetaka
                                        -- Povratak na izvršni deo programa NIJE MOGUĆ!
        /* NO_DATA_FOUND je predefinisani IZUZETAK */
        WHEN NO_DATA_FOUND THEN
            Raise_application_error (-20000, 'Deo mora biti sadržan u najmanje jednoj potklasi');
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;
```

Naredba za kreiranje serverskih funkcija

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

Naredba za kreiranje serverskih funkcija

- IN - specifikacija ulaznog parametra funkcije
  - vrednost parametra se zadaje pri pozivu funkcije i ne sme da se menja unutar funkcije
  - dozvoljeno je zadavanje DEFAULT vrednosti parametra
  - prenos parametra po referenci
- OUT - specifikacija izlaznog parametra funkcije
- IN OUT - specifikacija ulazno-izlaznog parametra funkcije
- **NAPOMENA:** Ne savetuje se da se u okviru funkcije deklarišu IN OUT, ili OUT parametri!
  - Ukoliko je to neophodno, treba funkciju preformulisati u proceduru!

Naredbe za menjanje i brisanje serverskih funkcija

```
ALTER FUNCTION [schema.]function_name COMPILE;
```

```
DROP FUNCTION [schema.]function_name;
```

Funkcije

- Obezbeđenje povratka vrednosti funkcije

  RETURN expression;

- Izraz expression mora biti kompatibilan s tipom povratnog podatka funkcije ret_datatype
- **NAPOMENA:** Svaka funkcija, u svom proceduralnom delu, ili delu za obradu izuzetaka, mora posedovati bar jednu naredbu RETURN

Primer kreiranja funkcije s deklaracijom ulaznih parametara

```
CREATE OR REPLACE FUNCTION F_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
      ) RETURN BOOLEAN
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;
      RETURN TRUE;
EXCEPTION
      WHEN OTHERS THEN
        ROLLBACK;
        RETURN FALSE;
END F_INS_Radnik;
```

Deklarisanje PL/SQL promenljivih i konstanti

identifier [CONSTANT] datatype

[NOT NULL] [:= | DEFAULT expr]

identifier [CONSTANT] {variable%TYPE | table.column%TYPE}

[NOT NULL] [:= | DEFAULT expr]

Deklarisanje PL/SQL promenljivih i konstanti

- Pravila deklarisanja:
  - Konstante moraju biti inicijalizovane.
  - NOT NULL promenljive moraju biti inicijalizovane.
  - Jedna deklaracija dozvoljava deklarisanje tačno jednog identifikatora.
  - Uvesti i poštovati konvencije imenovanja promenljivih i konstanti.
  - Ne nazivati promenljive i konstante istim imenima, kao što su nazivi kolona tabela, ili nazivi samih tabela.

Osnovne PL/SQL naredbe

- Naredba dodele vrednosti
  - Variable := expression

- Primeri upotrebe naredbe za dodelu vrednosti

```
DECLARE
    v_a BOOLEAN := TRUE;
    v_b NUMBER NOT NULL := 0;
BEGIN
    v_a := 5 > 3;
    v_b := v_b + 1;
END;
```

Dodela vrednosti varijabli korišćenjem SELECT naredbe

SELECT select_list

INTO variable[, variable]

FROM table

[WHERE condition]

...

Direktni način upotrebe SELECT naredbe

- SELECT naredba mora da vrati **JEDAN I SAMO JEDAN** red
- U protivnom, dolazi do pokretanja odgovarajućih izuzetaka
- Klauzula INTO obezbeđuje memorisanje vrednosti preuzete (selektovane) torke
- U izrazima, upotrebljenim u okviru naredbe SELECT, moguće je referenciranje na PL/SQL i bind (host) promenljive
- NAPOMENA: važno je poštovati konvencije imenovanja promenljivih, kolona tabela i samih tabela

Primer dodele vrednosti varijabli korišćenjem SELECT naredbe

```
DECLARE
    v_ime Radnik.Ime%TYPE;
    v_prz Radnik.Prz%TYPE;
BEGIN
      SELECT ime, prz
      INTO v_ime, v_prz
      FROM Radnik
      WHERE mbr = 20;
      ...
END;
```

Upotreba SQL naredbi u PL/SQL-u

- Dva načina upotrebe:
  - direktni
  - posredni, putem PL/SQL kursora

Direktni način upotrebe SELECT naredbe

```
SELECT select_list
INTO {variable[, variable]... | record_variable}
FROM table
[WHERE condition]
...
```

Primeri direktne upotrebe naredbe SELECT

```
DECLARE
    v_Count NUMBER(3);
BEGIN
    SELECT COUNT(*)
    INTO   v_Count
    FROM   Projekat;
    DBMS_OUTPUT.PUT_LINE(v_Count);
END;
```

Primeri direktne upotrebe naredbe SELECT

```
DECLARE
    V_Spr Projekat.Spr%TYPE := 10;
    V_Nap Projekat.Nap%TYPE;
    V_Nar Projekat.Nar%TYPE;
BEGIN
    SELECT Spr, Nap, Nar
    INTO   V_Spr, V_Nap, V_Nar
    FROM   Projekat
    WHERE Spr = V_Spr;
    DBMS_OUTPUT.PUT_LINE(v_Spr);
    DBMS_OUTPUT.PUT_LINE(v_Nap);
    DBMS_OUTPUT.PUT_LINE(v_Nar);
END;
```

Kursori u jezku PL/SQL

- Implicitni (SQL)
- Eksplicitni
  - Deklariše se programski
  - Njime se upravlja programski

Implicitni SQL kursor

- Sve SQL naredbe se parsiraju i izvršavaju u okviru kursorskih područja
- DML naredbama, koje se izvršavaju u PL/SQL bloku, dodeljuju se kursorska područja (kursori), čiji je programski naziv SQL
  - Implicitni SQL kursor
- Moguće je ispitivanje statusa implicitnog SQL kursora, nakon svake izvršene DML naredbe

Implicitni SQL kursor

- Funkcije ispitivanja statusa implicitnog SQL kursora
  - SQL%FOUND
    - TRUE, ako je bar jedan red bio predmet poslednje DML operacije, inače FALSE
  - SQL%NOTFOUND
    - TRUE, ako ni jedan red nije bio predmet poslednje DML operacije, inače FALSE
  - SQL%ROWCOUNT
    - broj redova, koji su bili predmet poslednje DML operacije
  - SQL%ISOPEN
    - uvek ima vrednost FALSE.
    - Upravljanje (otvaranje i zatvaranje) implicitnim kursorima je uvek automatsko. Neposredno nakon svake DML operacije, SQL kursorsko područje se automatski zatvori.

Primer

```
BEGIN
  UPDATE Projekat
  SET Nap = ''
  WHERE 1=2;

  DBMS_OUTPUT.PUT_LINE('Jedan update sa WHERE  USLOVOM 1=2');
  DBMS_OUTPUT.PUT_LINE(sql%rowcount || ' zapisa');
END;
```

Kursori u jezku PL/SQL

- Deklarisanje kursora

  CURSOR naziv_kursora [(lista_formalnih_parametara)] IS SELECT ...

- Otvaranje kursora

  OPEN naziv_kursora [(lista_stvarnih_parametara)];

- Preuzimanje torke kursora

  FETCH naziv_kursora INTO [var1, var2,... | record_var];

- Zatvaranje kursora

  CLOSE naziv_kursora;

Funkcije ispitivanja statusa eksplicitnog kursora

- naziv_kursora%FOUND
  - TRUE, ako je bar jedan red bio predmet poslednje fetch operacije, inače FALSE
- naziv_kursora%NOTFOUND
  - TRUE, ako ni jedan red nije bio predmet poslednje fetch operacije, inače FALSE
- naziv_kursora%ROWCOUNT
  - broj redova, koji su bili predmet poslednje fetch operacije
- naziv_kursora%ISOPEN
  - TRUE, ako je kursor otvoren, a inače FALSE

Eksplicitno deklarisani kursor

```
DECLARE
    Ukup_plt NUMBER;
    L_Mbr radnik.Mbr%TYPE;
    L_Plt radnik.Plt%TYPE;
    CURSOR spisak_rad                    -- eksplicitno deklarisani kursor
    IS
    SELECT Mbr, Plt
    FROM radnik
    WHERE Mbr BETWEEN 01 AND 99;
BEGIN
    Ukup_Plt := 0;
    OPEN spisak_rad;                     -- otvoren kursor, izvršava se SELECT
    LOOP
       FETCH spisak_rad INTO L_Mbr, L_Plt;
       EXIT WHEN spisak_rad%NOTFOUND;    -- uslov izlaska iz petlje
       Ukup_Plt := Ukup_Plt + L_Plt;
    END LOOP;
    CLOSE spisak_rad;                    -- zatvoren kursor
    DBMS_OUTPUT.PUT_LINE('Plata je: ' || Ukup_Plt);
END;
```

Primer kursora sa parametrima i %ROWCOUNT

```
DECLARE
    Ukup_plt NUMBER;
    L_tek_red radnik%ROWTYPE;
    CURSOR spisak_rad (D_gran radnik.Mbr%TYPE, G_gran radnik.Mbr%TYPE)
    IS                                          -- kursor, deklarisan s parametrima
    SELECT *
    FROM radnik
    WHERE Mbr BETWEEN D_gran AND G_gran;
BEGIN
    Ukup_Plt := 0;
    OPEN spisak_rad (01, 99);                   -- otvoren kursor, izvršava se SELECT
    LOOP
      FETCH spisak_rad INTO L_tek_red;
      EXIT WHEN (spisak_rad%NOTFOUND) OR (spisak_rad%ROWCOUNT > 5);
      Ukup_Plt := Ukup_Plt + L_tek_red.Plt;
    END LOOP;
    CLOSE spisak_rad;                           -- zatvoren kursor
    DBMS_OUTPUT.PUT_LINE('Plata je: ' || Ukup_Plt);
END;
```

Zadatak

Napisati PL/SQL blok koji će:

- Ispisati sve radnike kojima je šef Savo Oroz. Koristiti eksplicitni kursor za rešavanje zadatka.
- Ispis rezultata treba da izgleda na sledeći način:

Sef: _____

Ime zaposlenog je _____

Ime zaposlenog je _____

...

Rešenje

```
DECLARE
  v_mbr RADNIK.MBR%TYPE;
  v_tek_radnik RADNIK%ROWTYPE;
  CURSOR radnici(p_sef Radnik.MBR%TYPE)
  IS
  SELECT *
  FROM Radnik
  WHERE sef = p_sef;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Sef: Savo Oroz');
  SELECT MBR INTO v_mbr FROM Radnik
  WHERE IME LIKE 'Savo' and PRZ LIKE 'Oroz';

  OPEN radnici(v_mbr);
  LOOP
    FETCH radnici INTO v_tek_radnik;
    EXIT WHEN (radnici%NOTFOUND);
    DBMS_OUTPUT.PUT_LINE('Radnik: ' || v_tek_radnik.ime || ' ' || v_tek_radnik.prz);
  END LOOP;
  CLOSE radnici;
END;
```

Složeni tipovi podataka

- PL/SQL tip sloga
- PL/SQL tip kolekcije
  - INDEX BY tables – indeksirane tabele
  - nested tables – "ugnježdene" tabele
  - VARRAY – nizovi ograničene maksimalne dužine

PL/SQL tip sloga

- Deklarisanje

1. TYPE type_name IS RECORD (field_declaration[, field_declaration]...);

   <field_declaration>:

   field_name {field_type | variable%TYPE

   | table.column%TYPE | table%ROWTYPE}

   [[NOT NULL] {:= | DEFAULT} expr]

2. Identifier type_name;

PL/SQL tip sloga

- Referenciranje polja sloga

  identifier.field_name

%ROWTYPE atribut

- Deklariše promenljivu prema kolekciji kolona u tabeli ili pogledu baze podataka
- Ispred %ROWTYPE može da stoji ime tabele ili pogleda
- Polja u slogu imaju isti naziv i tip podatka kao i kolone u tabeli ili pogledu

Sintaksa

identifier table%ROWTYPE;

Primeri upotrebe promenljivih tipa sloga

```
DECLARE
    TYPE T_ProjSlog IS RECORD (
            Spr Projekat.Spr%TYPE := 10,
            Nap Projekat.Nap%TYPE
    );
    V_Proj T_ProjSlog;
BEGIN
    SELECT Spr, Nap
    INTO   V_Proj
    FROM   Projekat
    WHERE Spr = V_Proj.Spr;
    DBMS_OUTPUT.PUT_LINE('Naziv projekta je: ' || V_Proj.Nap );
END;
```

Primeri upotrebe promenljivih tipa sloga

```
DECLARE
    V_Proj Projekat%ROWTYPE;
BEGIN
    SELECT *
    INTO   V_Proj
    FROM   Projekat
    WHERE Spr = 10;
END;
```

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 deklarisati

Primer eksplicitno deklarisanog kursora 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;
```

Upotreba kursorske FOR petlje, s implicitno deklarisanim kursorom

```
DECLARE
  Ukup_Plt NUMBER;
BEGIN
     Ukup_Plt := 0;
     FOR p_tek_red IN (SELECT * FROM radnik -- otvoren kursor, izvršava SELECT
                         WHERE Mbr BETWEEN 01 AND 99) LOOP
     Ukup_Plt := Ukup_Plt + p_tek_red.Plt;
     END LOOP;   -- zatvoren kursor
     DBMS_OUTPUT.PUT_LINE('Plata je: ' || Ukup_Plt);
END;
```

acs
applied computer science
UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

DAIS
Data Science and
Information Systems Group

Kraj prezentacije

Serversko programiranje

Baze podataka