

# Sistemi baza podataka

---

*dr Vladimir Dimitrieski*

*Nikola Todorović*

*Jelena Hrnjak*

*Vladimir Jovanović*

---

# **PL/SQL – IMENOVANI BLOKOVI**

# Vrste PL/SQL blokova

---

- Neimenovani (anonimni) blok

```
[DECLARE
```

```
    ...                -- Deklarativni deo bloka
```

```
]
```

```
BEGIN
```

```
    ...                -- Izvršni deo bloka
```

```
[EXCEPTION
```

```
    ...                -- Deo bloka za obradu izuzetaka
```

```
]
```

```
END;
```

# Imenovani blokovi 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

# Imenovani blokovi u PL/SQL-u

---

- Vrste
  - procedura
    - predstavlja naredbu koja se poziva kao i bilo koja druga naredba – navođenjem naziva
    - **nema povratnu vrednost**
    - može da ima **ulazne i izlazne parametre**
  - funkcija
    - predstavlja unarni operator koji se koristi u izrazima i namenjen je da vrati izračunatu vrednost u izraz iz kojeg je pozvan
    - **ima povratnu vrednost**
    - može da ima **samo ulazne parametre**

# Imenovani blokovi u PL/SQL-u

---

## Zaglavlje\_programskog\_bloka

**IS | AS**

[ ... -- Deklarativni deo bloka (~DECLARE)  
]

**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, deklarirana unutar nekog PL/SQL bloka (programa)
- **Klijentska procedura ili funkcija**
  - procedura ili funkcija, deklarirana u okviru nekog alata iz Oracle Developer Suite
  - nalazi se i izvršava na srednjem sloju (aplikativnom serveru)

---

# **PL/SQL – PROCEDURE**



# 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 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 (-20001, '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 kljuka.');
```

```
    WHEN VALUE_ERROR THEN  
        ROLLBACK;  
        Raise_application_error (-20001, '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 procedure

---

```
CREATE OR REPLACE PROCEDURE A  
IS
```

```
    PROCEDURE B (P_1 IN NUMBER)
```

```
    IS
```

```
    BEGIN
```

```
        NULL;
```

```
    END;
```

```
BEGIN
```

```
    B(10);
```

```
END;
```

---

# **PL/SQL – FUNKCIJE**



# 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;
```

# Primer kreiranja funkcije s tipom povratnog podatka koji je tip sloga

---

```
CREATE OR REPLACE FUNCTION F_SEL_Projekat
(P_Spr IN Projekat.Spr%TYPE) RETURN Projekat%ROWTYPE
IS
    V_Proj Projekat%ROWTYPE;
BEGIN
    SELECT *
    INTO   V_Proj
    FROM   Projekat
    WHERE  Spr = P_Spr;
    RETURN V_Proj;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END F_SEL_Projekat;
```

## Primer kreiranja funkcije za izračunavanje zadatog procenta od zadatog broja

---

```
CREATE OR REPLACE FUNCTION F_PctInc  
    (P_Num IN NUMBER,  
     P_Pct IN NUMBER  
    ) RETURN NUMBER  
  
IS  
  
BEGIN  
    RETURN P_Num * (1 + P_Pct / 100);  
  
END F_PctInc;
```

# Pozivanje PL/SQL funkcije

---

- Pozivanje funkcije, kao unarnog operatora u izrazu

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



# Pozivanje PL/SQL funkcije

---

- 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.

# Moguća mesta (izrazi) u kojima se mogu pozivati serverske funkcije

---

- u bilo kojem PL/SQL ili SQL izrazu
  - u okviru naredbe dodele vrednosti
  - u okviru selekcione CASE funkcije
  - u okviru IF i LOOP naredbi
  - u okviru RETURN naredbe funkcije
  - kao stvarni parametar potprograma, na mestu formalnog parametra tipa IN

# Moguća mesta (izrazi) u kojima se mogu pozivati serverske funkcije

---

- U SQL izrazima u okviru SQL naredbi
  - Naredba SELECT
    - u SELECT listi
    - u klauzulama WHERE, GROUP BY, HAVING, ORDER BY, CONNECT BY START WITH
  - Naredba INSERT
    - u klauzuli VALUES
  - Naredba UPDATE
    - u klauzuli SET
    - u klauzuli WHERE
  - Naredba DELETE
    - u klauzuli WHERE

# Moguća mesta (izrazi) u kojima se mogu pozivati serverske funkcije

---

- **NAPOMENA:** Ograničenja u pisanju funkcija, kada se funkcije koriste u SQL izrazima naredbe SELECT, ili DML naredbi
  - Mora se koristiti pozicioni način pozivanja
  - Ne smeju se koristiti tipovi povratnih podataka, specifični samo za PL/SQL (BOOLEAN, %ROWTYPE, %TYPE, RECORD)
  - Ne smeju se deklarirati formalni parametri tipa IN OUT ili OUT
  - Ne smeju se koristiti naredbe SAVEPOINT, COMMIT i ROLLBACK
  - Ne smeju se koristiti DML naredbe, ako je funkcija u SELECT naredbi
  - Ne smeju se koristiti ni DML naredbe ni SELECT naredba nad istom tabelom, ili povezanim tabelama, ako je funkcija u DML naredbi
  - Funkcije koje zove takva funkcija, takođe moraju zadovoljiti iste uslove

# Primeri mogućih načina pozivanja prethodno kreiranih funkcija

---

```
BEGIN
```

```
  IF F_INS_Radnik(200, 'Antic', 'Ante', 20000,
```

```
  TO_DATE('01.10.1965', 'DD.MM.YYYY'), 2000) THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Uspesan unos. Transakcija  
    potvrđena.');
```

```
  ELSE
```

```
    DBMS_OUTPUT.PUT_LINE('Neuspesan unos. Transakcija  
    ponistena.');
```

```
  END IF;
```

```
END;
```

```
DECLARE
```

```
  V_Uspeh BOOLEAN;
```

```
BEGIN
```

```
  V_Uspeh := F_INS_Radnik( P_Ime => 'Danko', P_Prz => 'Danic',  
  P_Mbr=> 250, P_Plt => 21000);
```

```
END;
```

# Primeri mogućih načina pozivanja prethodno kreiranih funkcija

---

```
ACCEPT P_Spr PROMPT 'Unesi sifru projekta'
DECLARE
    V_Proj Projekat%ROWTYPE;
BEGIN
    V_Proj := F_SEL_Projekat(&P_Spr);
    DBMS_OUTPUT.PUT_LINE('Projekat: ' || V_Proj.Spr || ' ' ||
        V_Proj.Nap);
END;
```

```
ACCEPT P_Spr PROMPT 'Unesi sifru projekta'
DECLARE
    V_Proj Projekat%ROWTYPE;
BEGIN
    V_Proj := F_SEL_Projekat(P_Spr => &P_Spr);
    DBMS_OUTPUT.PUT_LINE('Projekat: ' || V_Proj.Spr || ' ' ||
        V_Proj.Nap);
END;
```

# Primeri mogućih načina pozivanja prethodno kreiranih funkcija

---

```
SELECT Mbr, Prz, Ime, F_PctInc(Plt, 10)  
FROM Radnik  
WHERE Plt <= 9000;
```

# Deklarisanje i pozivi lokalnih funkcija

---

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



# Deklarisanje i pozivi lokalnih funkcija

---

```
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;
```

# Primer deklaracije i poziva lokalne funkcije

---

```
CREATE OR REPLACE PROCEDURE A
  V_X VARCHAR2(1);
IS
  FUNCTION B (P_1 IN NUMBER) RETURN VARCHAR2
  IS
  BEGIN
    RETURN 'B';
  END;
BEGIN
  V_X := B(10);
END;
```

# Zadatak

---

- Napisati lokalnu proceduru koja će, u okviru jedne transakcije, putem kursora, preuzimati, redom, sve torke iz tabele Radnik 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_Rad_zad1 IS TABLE OF radnik%rowtype INDEX BY BINARY_INTEGER;
Tabela T_Rad_zad1;
i integer;
PROCEDURE P_Radnik_Zad1(Tabela OUT NOCOPY T_Rad_zad1)
IS
    i BINARY_INTEGER:=0;
BEGIN
FOR rec IN (SELECT * FROM Radnik) LOOP
    Tabela(i):=rec;
    i:=i+1;
END LOOP;
END;
begin
P_Radnik_Zad1(Tabela);
i:= Tabela.FIRST;
WHILE i<=Tabela.LAST LOOP
    DBMS_OUTPUT.PUT_LINE('Naziv ime radnika: ' || Tabela(i).ime || ' ' || Tabela(i).prz);
    DBMS_OUTPUT.PUT_LINE(Datum rodjenja radnika: ' || Tabela(i).god);
    DBMS_OUTPUT.PUT_LINE(Plata radnika: ' || Tabela(i).plt);
    i:=Tabela.NEXT(i);
END LOOP;
end;
```

# Zadatak za vežbu

---

- Napisati funkciju koja će, u okviru jedne transakcije, putem kursora, preuzimati, redom, sve torke iz tabele Radnik, uređene u opadajućem redosledu matičnog broja, i prebacivati ih u PL/SQL tabelarnu kolekciju. Uz svaku preuzetu torku iz tabele Radnik, treba u okviru elementa te kolekcije, inicijalizovati novu kolekciju koja će sadržati skup svih naziva projekata na kojima ti radnici rade. Takođe, treba za svakog radnika prikazati i nazive projekata kojima rukovodi. Nazive projekata na kojima radnik radi i kojima rukovodi, treba selektovati putem posebnih kursora.
- Tabelarna kolekcija selektovanih radnika treba da predstavlja izlazni podatak funkcije.
- Proveriti ispravnost rada funkcije pozivima na konkretnim primerima.

---

# **PL/SQL – DINAMIČKI SQL**

# Dinamički SQL

---

- Dinamički SQL omogućava izvršavanje (PL/)SQL naredbi koje su tek poznate u fazi izvršavanja (*runtime, late binding*)
  - na primer, nazivi tabela su poznati tek u fazi izvršavanja programa
- Oracle DBMS pruža dva načina za rad sa dinamičkim SQL naredbama u PL/SQL programima:
  - Pozivanje nativnog dinamičkog SQL-a, naredbe se direktno pišu u PL/SQL blokove.
  - Pozivanje procedura iz DBMS\_SQL paketa.
    - Obavezno za naredbe sa nepoznatim brojem ulaznih ili izlaznih parametara

# Dinamički SQL

---

```
plsql_block := 'BEGIN calc_stats(:x, :x, :y, :x); END;';  
EXECUTE IMMEDIATE plsql_block  
    USING a, b; -- calc_stats(a, a, b, a)
```

```
query_str := 'SELECT COUNT(*) FROM ' || ' emp_' || loc || ' WHERE job =  
            :job_title';  
EXECUTE IMMEDIATE query_str  
    INTO num_of_employees  
    USING job;
```



# Dinamički SQL

---

```
CREATE OR REPLACE PROCEDURE query_invoice( month VARCHAR2, year
VARCHAR2)
IS
    TYPE cur_typ IS REF CURSOR;
    c cur_typ;
    query_str VARCHAR2(200);
    inv_num NUMBER := 1;
    inv_cust VARCHAR2(20);
    inv_amt NUMBER;
BEGIN
    query_str := 'SELECT num, cust, amt FROM inv_' || month || '_' || year || ' WHERE
invnum = :id';
    OPEN c FOR query_str USING inv_num;
    LOOP FETCH c INTO inv_num, inv_cust, inv_amt;
        EXIT WHEN c%NOTFOUND; -- process row here
    END LOOP;
    CLOSE c;
END;
```

# Zadatak za vežbu

---

- Napisati funkciju 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 funkcije. Obezbediti programsko (dinamičko) formiranje SELECT naredbe kursorskog područja. Selektovane torke treba preneti u tabelarnu promenljivu (kolekciju). Tabelarna kolekcija selektovanih radnika treba da predstavlja izlazni podatak funkcije.

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 funkcije pozivima na konkretnim primerima.