

# Sistemi baza podataka

---

*Slavica Aleksić*

*slavica@uns.ns.ac.yu*

# Rad u učionici

---

- Podaci potrebni za konektovanje na bazu podataka (db2013)
  - Host Name:
    - 192.168.0.102
  - Port Number: 1521
  - Oracle SID, database name:
    - db2013

# Prikaz vrednosti izraza

---

- PL/SQL na nivou DBMS-a i SQL\*Plus-a – kombinacija:
  - SET SERVEROUTPUT ON i
  - DBMS\_OUTPUT.PUT\_LINE (message)
- SQL Developer
  - View -> Dbms Output
  - +

# Primeri predaje vrednosti izraza

---

```
DECLARE
```

```
    V_A NUMBER;
```

```
    S_A NUMBER := '10';
```

```
BEGIN
```

```
    V_A := S_A * 6/1.5;
```

```
    DBMS_OUTPUT.PUT_LINE('Stampa vrednosti za V_A');
```

```
    DBMS_OUTPUT.PUT_LINE('Vrednost za V_A je: ' ||  
V_A);
```

```
    DBMS_OUTPUT.PUT_LINE('Vrednost za V_A je: ' ||  
TO_CHAR(V_A));
```

```
END;
```

# Ugrađivanje blokova i opseg delovanja promenljivih

---

```
DECLARE
```

```
-- opseg delovanja x – do kraja spoljnjeg bloka
```

```
  x BINARY_INTEGER;
```

```
BEGIN
```

```
  DECLARE
```

```
-- opseg delovanja y – do kraja unutrašnjeg bloka
```

```
  y PLS_INTEGER;
```

```
  BEGIN
```

```
    y := x;
```

```
  END;
```

```
END;
```

# Ugrađivanje blokova i opseg delovanja promenljivih

---

- **NAPOMENA:** Naziv lokalno deklarisanе konstrukcije ima prioritet, u odnosu na naziv globalno deklarisanе konstrukcije

# Ugrađivanje blokova i opseg delovanja promenljivih

---

```
DECLARE
```

```
  x BINARY_INTEGER;    -- vidljivost: u spolnjem bloku
```

```
BEGIN
```

```
  DECLARE
```

```
    x VARCHAR2(20);
```

```
      -- vidljivost: samo u unutrašnjem bloku
```

```
    y PLS_INTEGER;
```

```
      -- vidljivost: samo u unutrašnjem bloku
```

```
BEGIN
```

```
  y := TO_NUMBER(x, '$99,990.00');
```

```
END;
```

```
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]  
...
```

# 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

# 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;
```

# Implicitni SQL kursor

---

- Sve SQL naredbe se parsiraju i izvršavaju u okviru kursorских područja
- DML naredbama, koje se izvršavaju u PL/SQL bloku, dodeljuju se kursorска područja (kursori), čiji je programски 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 kursorско 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;
```

# DML naredbe

---

- Normalna upotreba naredbi INSERT, UPDATE i DELETE



# Primeri upotrebe DML naredbi

---

```
ACCEPT D_Prz PROMPT 'Unesite prezime: '  
ACCEPT D_Ime PROMPT 'Unesite ime: '
```

```
BEGIN
```

```
    INSERT INTO Radnik (Mbr, Prz, Ime, God)  
    VALUES (SEQ_Mbr.NEXTVAL, '&D_Prz', '&D_Ime',  
    SYSDATE);
```

```
    IF SQL%FOUND THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Dodata nova toraka u  
tabelu Radnik.');
```

```
    ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('Unos torke u tabelu  
Radnik nije uspeo.');
```

```
    END IF;
```

```
END;
```

# Primeri upotrebe DML naredbi

---

```
DECLARE
```

```
  v_Mbr radnik.mbr%TYPE := 203;
```

```
  broj_del NUMBER;
```

```
BEGIN
```

```
  DELETE FROM radnik
```

```
  WHERE  mbr = v_Mbr;
```

```
  broj_del := SQL%ROWCOUNT;
```

```
  DBMS_OUTPUT.PUT_LINE('Obrisano je: ' ||  
    broj_del || ' radnika');
```

```
END;
```

# Naredbe za upravljanje tokom izvođenja programa

---

- Naredba selekcije
- Naredbe iteracije

# Naredba selekcije

---

```
IF logički_izraz THEN
    blok_izvršnih_naredbi;
[ELSIF logički_izraz THEN
    blok_izvršnih_naredbi;
]...
[
ELSE
    blok_izvršnih_naredbi;
]
END IF;
```

# Naredbe iteracije

---

- Bezuslovna (beskonačna) iteracija / LOOP

```
LOOP
```

```
    blok_izvršnih_naredbi;
```

```
END LOOP;
```

- Uslovna iteracija, s testom uslova na početku /  
WHILE LOOP

```
WHILE logički_izraz LOOP
```

```
    blok_izvršnih_naredbi;
```

```
END LOOP;
```

# Naredbe iteracije

---

- Brojačka iteracija / FOR LOOP  
FOR brojač IN [REVERSE]  
    donja\_granica..gornja\_granica LOOP  
    blok\_izvršnih\_naredbi;  
END LOOP;

**NAPOMENA:** Brojačku promenljivu *brojač* nije potrebno deklarisati.

Korak brojača je uvek 1.

# Izlazak iz petlje / EXIT

---

- EXIT [labela] [WHEN logički\_izraz]
- EXIT se, najčešće, koristi u kombinaciji s безусловnom petljom LOOP ... END LOOP
  - Obezbeđenje formiranja uslovne petlje, s mogućnošću testa uslova petlje na bilo kojoj poziciji u petlji

<<labela>>

LOOP

...

EXIT [labela] [WHEN logički\_izraz]

...

END LOOP;

# Primeri upotrebe konstrukcija za upravljanje tokom izvođenja programa

---

```
BEGIN
```

```
  FOR i IN REVERSE 1..3 LOOP
```

```
    DBMS_OUTPUT.PUT_LINE('Vrednost brojaca i je: '  
  || TO_CHAR(i));
```

```
  END LOOP;
```

```
END;
```

```
BEGIN
```

```
  FOR i IN 1..3 LOOP
```

```
    DBMS_OUTPUT.PUT_LINE('Vrednost brojaca i je: '  
  || TO_CHAR(i));
```

```
  END LOOP;
```

```
END;
```



# Primeri upotrebe konstrukcija za upravljanje tokom izvođenja programa

---

```
DECLARE
```

```
    i NUMBER(1) := 1;
```

```
BEGIN
```

```
    WHILE i <= 3 LOOP
```

```
        DBMS_OUTPUT.PUT_LINE('Vrednost  
brojaca i je: ' || TO_CHAR(i));
```

```
        i := i + 1;
```

```
    END LOOP;
```

```
END;
```

# Primeri upotrebe konstrukcija za upravljanje tokom izvođenja programa

---

```
DECLARE
```

```
  i NUMBER(1) := 1;
```

```
BEGIN
```

```
  LOOP
```

```
    EXIT WHEN i > 3;
```

```
    DBMS_OUTPUT.PUT_LINE('Vrednost  
brojaca i je: ' || TO_CHAR(i));
```

```
    i := i + 1;
```

```
  END LOOP;
```

```
END;
```

# Primeri upotrebe konstrukcija za upravljanje tokom izvođenja programa

---

```
DECLARE
```

```
    i NUMBER(1) := 0;
```

```
BEGIN
```

```
    LOOP
```

```
        i := i + 1;
```

```
        DBMS_OUTPUT.PUT_LINE('Vrednost  
brojaca i je: ' || TO_CHAR(i));
```

```
        EXIT WHEN i >= 3;
```

```
    END LOOP;
```

```
END;
```

# Primeri upotrebe konstrukcija za upravljanje tokom izvođenja programa

---

```
ACCEPT N PROMPT 'N: '  
BEGIN  
  FOR i IN 1..&N LOOP  
    IF MOD(i, 2) = 0 THEN  
      DBMS_OUTPUT.PUT_LINE(i || ' je paran  
broj.);  
    ELSIF MOD(i, 2) = 1 THEN  
      DBMS_OUTPUT.PUT_LINE(i || ' je neparan  
broj.);  
    ELSE  
      DBMS_OUTPUT.PUT_LINE('Nemoguc  
slucaj.);  
    END IF;  
  END LOOP;  
END;
```

# Zadatak

---

Napisati PL/SQL blok koji će:

- interaktivno prihvatiti vrednosti za Mbr, Prz, Ime, Sef, Plt i God,
- dodati novu torku u tabelu Radnik, s prethodno preuzetim podacima i
- angažovati novododatog radnika na projektu sa Spr = 10 i 5 sati rada.

# Rešenje

---

BEGIN

```
INSERT INTO radnik (Mbr, Prz, Ime, Plt, God)
VALUES (&&Mbr, '&&Prz', '&&Ime', &&Plt,
'&&God');
```

```
INSERT INTO radproj (Mbr, Spr, Brc)
VALUES (&&Mbr, 10, 5);
```

```
COMMIT;
```

END;

# Zadatak

---

Napisati PL/SQL blok koji će:

- izbrisati angažovanje prethodno dodatog radnika na projektu sa šifrom 10 i obavestiti porukom korisnika da li je brisanje uspešno obavljeno,
- izbrisati prethodno dodatog radnika iz evidencije i obavestiti porukom korisnika da li je brisanje uspešno obavljeno,
- sačuvati vrednost za Mbr izbrisanog radnika u lokalnoj promenljivoj pod nazivom *Del\_Mbr*

# Rešenje

---

```
ACCEPT v_Mbr PROMPT 'MBR = '  
  
DECLARE  
    Del_Mbr radnik.Mbr%TYPE;  
BEGIN  
    DELETE FROM radproj  
    WHERE Mbr = &v_Mbr AND Spr = 10;  
    IF SQL%FOUND THEN  
        DBMS_OUTPUT.PUT_LINE('Brisanje rada na projektu uspesno obavljeno.');    ELSE  
        DBMS_OUTPUT.PUT_LINE('Brisanje rada na projektu nije uspesno obavljeno.');    END IF;  
  
    DELETE FROM radnik  
    WHERE Mbr = &v_Mbr ;  
    IF SQL%FOUND THEN  
        DBMS_OUTPUT.PUT_LINE('Brisanje radnika uspesno obavljeno.');    ELSE  
        DBMS_OUTPUT.PUT_LINE('Brisanje radnika nije uspesno obavljeno.');    END IF;  
    Del_Mbr := &Mbr;  
END;
```



# Zadatak

---

Kreirati tabelu Spisak\_zarada, korišćenjem SQL komande:

```
CREATE TABLE Spisak_zarada (Mbr NUMBER(3),  
Plt NUMBER(10, 2), Evri VARCHAR2(10),  
CONSTRAINT Sz_PK PRIMARY KEY (Mbr))
```

Napisati PL/SQL blok koji će:

za svaku torku iz tabele Radnik, za koju je matični broj u intervalu od 10 do 100, izuzimajući radnika s matičnim brojem 90, preneti u tabelu Spisak\_zarada matični broj, iznos plate, i inicijalizovati polje Evri sa vrednošću plate u evrima. Ukoliko radnik već postoji u tabeli izvršiti izmenu vrednosti obeležja Plt i Evri. Kurs evra treba da zadaje korisnik iz okruženja.

# Rešenje

---

```
ACCEPT E PROMPT 'Kurs evra je: '  
DECLARE  
    v_Plt Spisak_zarada.Plt%TYPE;  
    broj NUMBER :=0;  
BEGIN  
    FOR i IN 1..10 LOOP  
        IF i != 9 THEN  
            SELECT Plt INTO v_Plt FROM Radnik  
            WHERE Mbr = 10*i;  
            SELECT COUNT(*) INTO broj FROM Spisak_zarada  
            WHERE Mbr = 10*i;  
            IF broj = 0 THEN  
                INSERT INTO Spisak_zarada (Mbr, Plt, Evri)  
                VALUES (10*i, v_Plt, v_Plt*&E );  
            ELSE  
                UPDATE Spisak_zarada  
                SET Plt = v_Plt,  
                Evri = v_Plt*&E  
                WHERE Mbr = 10*i;  
            END IF;  
        END IF;  
    END LOOP;  
END;
```

# Zadatak

---

Napisati PL/SQL blok koji će:

Proveravati ima li radnika sa platom manjom od zadate. Ako ima povećati premiju za 20% svakom radniku koji ima takvu platu. Ukoliko radnik nema uopste premiju dodeliti mu premiju od 5000. Ako svi radnici imaju platu veću od zadate ispisati poruku o tome.

# Rešenje

---

```
ACCEPT plata PROMPT 'Plata = '  
DECLARE  
    broj_rad NUMBER;  
BEGIN  
    SELECT COUNT(*) INTO broj_rad FROM Radnik  
    WHERE Plt < &plata;  
    IF broj_rad = 0 THEN  
        DBMS_OUTPUT.PUT_LINE('Svi imaju platu vecu od ' ||  
        TO_CHAR(&plata));  
    ELSE  
        UPDATE Radnik  
        SET Pre = NVL(Pre*1.2,5000)  
        WHERE Mbr IN (SELECT Mbr FROM Radnik  
        WHERE Plt < &plata);  
    END IF;  
END;
```

# Kursori u jezku PL/SQL

---

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

# 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 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

# Primer eksplicitno deklarisanog kursora

---

```
DECLARE
    Ukup_plt NUMBER;
    L_Mbr radnik.Mbr%TYPE;
    L_Plt radnik.Plt%TYPE;

    CURSOR spisak_rad IS          -- eksplicitno deklarisan kursor
    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 eksplicitno deklarisanog kursora s parametrima i funkcijom %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 koji rade u sektorima kojima rukovode Pera Perić, Savo Oroz i Đoka Đokić. Ukoliko neko od njih nije rukovodilac obavestiti korisnika o tome.
- Ispis rezultata treba da izgleda na sledeći način:

Radnici kojima je sef ...

Ime zaposlenog je ...

Ime zaposlenog je ...

Radnici kojima je sef ...

Ime zaposlenog je ...

# Rešenje

---

```
DECLARE
CURSOR radnik_sef (p_sef in number)
IS SELECT * FROM radnik WHERE Sef = p_sef;

v_radnik Radnik%rowtype;
v_mbr Radnik.mbr%type := 0;
v_broj NUMBER;
BEGIN
SELECT mbr INTO v_mbr FROM radnik WHERE ime='Pera' and prz='Peric';
SELECT COUNT(*) INTO v_broj FROM radnik WHERE sef=v_mbr;
IF v_broj = 0 THEN
  DBMS_OUTPUT.PUT_LINE('Pera Peric nije sef');
ELSE
  DBMS_OUTPUT.PUT_LINE('Radnici kojima je sef Pera Peric');
  DBMS_OUTPUT.PUT_LINE("");
  OPEN radnik_sef(v_mbr);
  LOOP
    FETCH radnik_sef INTO v_radnik;
    EXIT WHEN radnik_sef%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Ime zaposlenog ' || ' je ' || v_radnik.ime || ' ' || v_radnik.prz);
  END LOOP;
  CLOSE radnik_sef;
END IF;
DBMS_OUTPUT.PUT_LINE("");
```

-- nastavak na sledecoj strani

# Rešenje

---

```
SELECT mbr INTO v_mbr FROM radnik WHERE ime='Savo' and prz='Oroz';
SELECT COUNT(*) INTO v_broj FROM radnik WHERE sef=v_mbr;
IF v_broj = 0 THEN
  DBMS_OUTPUT.PUT_LINE('Savo Oroz nije sef');
ELSE
  DBMS_OUTPUT.PUT_LINE('Radnici kojima je sef Savo Oroz');
  DBMS_OUTPUT.PUT_LINE("");
  OPEN radnik_sef(v_mbr);
  LOOP
    FETCH radnik_sef INTO v_radnik;
    EXIT WHEN radnik_sef%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Ime zaposlenog ' || ' je ' || v_radnik.ime || ' ' || v_radnik.prz);
  END LOOP;
  CLOSE radnik_sef;
END IF;
DBMS_OUTPUT.PUT_LINE("");
SELECT mbr INTO v_mbr FROM radnik WHERE ime='Djoka' and prz='Djokic';
SELECT COUNT(*) INTO v_broj FROM radnik WHERE sef=v_mbr;
IF v_broj = 0 THEN
  DBMS_OUTPUT.PUT_LINE('Djoka Djokic nije sef');
ELSE
  DBMS_OUTPUT.PUT_LINE('Radnici kojima je sef Djoka Djokic');
  DBMS_OUTPUT.PUT_LINE("");
  OPEN radnik_sef(v_mbr);
  LOOP
    FETCH radnik_sef INTO v_radnik;
    EXIT WHEN radnik_sef%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Ime zaposlenog ' || ' je ' || v_radnik.ime || ' ' || v_radnik.prz);
  END LOOP;
  CLOSE radnik_sef;
END IF;
END;
```

# Zadatak – primer ugnježenih kursora

---

Napisati PL/SQL blok koji će:

- Ispisati sve radnike po sektorima sa određenim šefom

# Rešenje

---

```
DECLARE
  CURSOR sefovi
  IS SELECT DISTINCT sef FROM radnik WHERE Sef IS NOT NULL;
  CURSOR radnici (p_sef in number)
  IS SELECT * FROM radnik WHERE sef = p_sef;
  v_sef Radnik.sef%type;
  v_ime Radnik.ime%type;
  v_prz Radnik.prz%type;
  v_radnik Radnik%rowtype;

BEGIN
  OPEN sefovi;
  LOOP
    FETCH sefovi INTO v_sef;
    EXIT WHEN sefovi%NOTFOUND;
    SELECT ime, prz INTO v_ime, v_prz FROM radnik WHERE Mbr = v_sef;
    DBMS_OUTPUT.PUT_LINE("");
    DBMS_OUTPUT.PUT_LINE('Radnici kojima je sef ' || v_ime || ' ' || v_prz);
    DBMS_OUTPUT.PUT_LINE("");
    OPEN radnici(v_sef);
    LOOP
      FETCH radnici INTO v_radnik;
      EXIT WHEN radnici%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Ime zaposlenog ' || ' je ' || v_radnik.ime || ' ' || v_radnik.prz);
    END LOOP;
    CLOSE radnici;
  END LOOP;
  CLOSE sefovi;
END;
```

# Zadatak

---

Napisati PL/SQL blok koji će:

za zadati naziv projekta, za svakog radnika koji radi na tom projektu i ima broj časova rada veći od jedan povećati premiju za 10 posto. Ako radnik uopšte nema premiju dati mu premiju od 1000.

# Rešenje

---

```
ACCEPT v_NazivProj CHAR PROMPT 'Unesite naziv projekta'
DECLARE
  CURSOR radnici (p_spr in number)
  IS SELECT r.mbr FROM radnik r, radproj rp WHERE r.mbr = rp.mbr
     AND rp.spr = p_spr AND rp.brc >1;

v_mbr Radnik.mbr%type;
v_spr Projekat.spr%type;
BEGIN
  SELECT spr INTO v_spr from projekat where nap = '&v_NazivProj';
  OPEN radnici(v_spr);
  LOOP
    FETCH radnici INTO v_mbr;
    EXIT WHEN radnici%NOTFOUND;
    UPDATE radnik
    SET pre = NVL(pre*1.1,1000)
    WHERE mbr = v_mbr;
  END LOOP;
  CLOSE radnici;
END;
```