

# Sistemi baza podataka

---

*Dr Ivan Luković*

*Dr Slavica Kordić*

*Vladimir Dimitrieski*

*Nikola Todorović*

*Milica Todorović*



---

# **PL/SQL - OSNOVI**

---

# 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

# Osnovna struktura PL/SQL bloka

---

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

# 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 višelinijuskog 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;

    -- Deo za obradu izuzetaka
EXCEPTION    -- 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;
```

# Osnovni leksički elementi

---

- **Skup simbola**

- A-Z, a-z, 0-9, (, ), &, <, >, =, !, :, ., ', @, %, ^, {, }, [, ], \_, ", #, ?, +, -, \*, /

- **Delimiteri**

- (, ), %, <>, !=, \*, \*\*, :=, itd.

- **Literali**

- numerički (114, 12.5, -1.E3)
- karakter ('O"vo je string')
- logički (TRUE, FALSE, NULL)

- **Komentari:**

- jednolinijski (--)
- višelinijski (/\* \*/)

- **Identifikatori**

- do 30 znakova, prvi znak mora biti slovo.
- dozvoljeni karakteri: slova, brojevi, \_, #, \$.



# Ugrađivanje blokova i tok izvršenja programa

---

DECLARE

Deklarativni deo programa:

\* GLOBALNE DEKLARACIJE

\* deklaracije procedura i funkcija

PROCEDURE | FUNCTION

lokalne deklaracije

BEGIN

EXCEPTION

END;

BEGIN

Izvršni deo programa:

DECLARE

lokalne deklaracije

BEGIN

EXCEPTION

END;

EXCEPTION

Deo za obradu izuzetaka:

\* WHEN <izuzetak> THEN

DECLARE

lokalne deklaracije

BEGIN

EXCEPTION

END;

END;

# Ugrađivanje blokova i tok izvršenja programa

---

- Važe uobičajeni mehanizmi toka izvršenja programa
  - Nakon završetka ugrađenog bloka, kontrola izvođenja programa se predaje pozivajućem ("okružujućem") bloku
- Koncept lokalnosti i globalnosti deklaracija važi na uobičajen način
  - Lokalne deklaracije nisu vidljive u pozivajućem bloku. Globalne deklaracije su vidljive u pozvanom bloku.

# Tipovi podataka

---

- Skalarni (osnovni)
  - Specifični Oracle tipovi i ANSI SQL standardni tipovi
  - karakter
    - VARCHAR2 (do 32767 bajtova)
    - CHAR (do 32767, default 1)
    - LONG (do 32760)
      - slično sa *varchar2* samo malo kraći
    - NVARCHAR2, NCHAR
  - numerički
    - NUMBER(p,s) (decimalni tip, od 1E-130 do 10E125 )
    - BINARY\_INTEGER (4B integer, u rasponu od  $-(2^{31}-1)$  do  $2^{31}-1$ )
    - PLS\_INTEGER (4B "pakovani" integer, u rasponu od  $-(2^{31}-1)$  do  $2^{31}-1$ ) – zauzima manje prostora
      - u novijim aplikacijama se često koristi umesto *binary\_integer*-a zbog boljih performansi

# Tipovi podataka

---

- Skalarni (osnovni)
  - datumski
    - DATE
    - TIMESTAMP
    - TIMESTAMP WITH TIME ZONE
    - TIMESTAMP WITH LOCAL TIME ZONE
    - INTERVAL DAY TO SECOND
    - INTERVAL YEAR TO MONTH
  - logički
    - BOOLEAN

# Tipovi podataka

---

- Složeni (Composite)
  - RECORD
  - TABLE
  - VARRAY
- Pokazivački (Reference)
  - REF CURSOR, REF objektni\_tip
  - ROWID, RAW
- LOB
  - BFILE (fajl do 4GB)
  - BLOB (do 4GB)
  - CLOB (do 4GB), NCLOB (do 4GB)
  - RAW (do 32760), LONG RAW (do 32760)

# Promenljive i konstante

---

- PL/SQL promenljive i konstante
  - Skalarne (osnovne)
  - Složene (Composite)
  - Pokazivačke (Reference)
  - LOB

# Deklarisanje PL/SQL promenljivih i konstanti

---

identifier [CONSTANT] datatype  
[NOT NULL] [:= | DEFAULT expr]

identifier [CONSTANT] {variable%TYPE |  
table.column%TYPE}  
[NOT NULL] [:= | DEFAULT expr]

# Primeri deklaracija promenljivih i konstanti

---

DECLARE

V\_prom1 NUMBER(2);

V\_prom2 CHAR;

V\_prom3 VARCHAR2(40) := ' ';

V\_prom4 VARCHAR2(40) NOT NULL := ' ';

V\_prom5 VARCHAR2(40) **NOT NULL DEFAULT** ' ';

V\_prom6 DATE NOT NULL := SYSDATE + 2;

C\_prom7 **CONSTANT** DATE:= SYSDATE;

V\_prom8 **V\_Prom6%TYPE** := TO\_DATE('01.01.2001',  
'DD.MM.YYYY');

V\_prom9 **Radnik.Mbr%TYPE** := 100;

BEGIN

**NULL;**

END;



# 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.**
    - obično *v\_imeVariable*

# PL/SQL izrazi

---

- Klasifikacija PL/SQL izraza:
  - Numerički izrazi
  - Karakter izrazi
  - Logički izrazi
  - Datumski izrazi
  - Selekcioni izrazi (izrazi IF-tipa)

# PL/SQL izrazi

---

- Izrazi se formiraju na uobičajen način, korišćenjem odgovarajućih operatora.
- U izrazima je dozvoljena upotreba najvećeg broja predefinisanih ***jednosložnih*** ORACLE SQL funkcija.
- U izrazima je dozvoljena upotreba jednosložnih, korisnički definisanih funkcija.
- Dozvoljena je upotreba predefinisanih operatora: [NOT] IN, [NOT] LIKE, [NOT] BETWEEN AND i IS [NOT] NULL.
- **LOGIČKI TIP PODATAKA:** BOOLEAN. Moguće vrednosti: TRUE, FALSE, NULL. **Vrednost NULL se u logičkim izrazima tretira kao FALSE!**

# PL/SQL izrazi

---

- Konverzija podataka različitih tipova
  - Pri izračunavanju izraza, vrši se implicitna konverzija podataka, kada je to moguće.
  - Preporučljivo je koristiti uvek funkcije za eksplicitnu konverziju podataka TO\_CHAR, TO\_DATE i TO\_NUMBER.

## Selekcioni izrazi (Izrazi IF tipa)

---

```
CASE [expr] WHEN comparison_expr1 THEN return_expr1
      [ WHEN comparison_expr2 THEN return_expr2
        WHEN comparison_exprn THEN return_exprn
      ]
      [ ELSE else_expr
      ]
END;
```

# Primer selekcionog PL/SQL izraza

---

CASE Status

WHEN 'A' THEN 'Odlican'

WHEN 'B' THEN 'Zadovoljava'

ELSE 'Ne zadovoljava'

END;

CASE

WHEN Status = 'A' THEN 'Odlican'

WHEN Status = 'B' THEN 'Zadovoljava'

ELSE 'Ne zadovoljava'

END;



# Osnovne PL/SQL naredbe

---

- **"Prazna" naredba**
  - **NULL**
  - svaka BEGIN – END sekcija mora da ima makar jednu naredbu
- Primer upotrebe prazne naredbe

```
BEGIN  
  NULL;  
END;
```

# 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;
```





# Prikaz vrednosti izraza

---

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

**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;
```



# Zadatak

---

- U konzolu ispisati trenutni datum, datum rođenja, dan u sedmici kada ste rođeni kao i broj dana koje ste proživeli do sada. Datum rođenja uneti kroz interaktivni prompt.

# Rešenje

---

```
BEGIN
```

```
  DBMS_OUTPUT.PUT_LINE('Danas je: ' || SYSDATE);
```

```
  DBMS_OUTPUT.PUT_LINE('Rodjen sam: ' || TO_DATE('&Dat_rodj',  
    'DD.MM.YYYY'));
```

```
  DBMS_OUTPUT.PUT_LINE('Bio je: ' || TO_CHAR(TO_DATE(  
    '&Dat_rodj', 'DD.MM.YYYY'), 'DAY'));
```

```
  DBMS_OUTPUT.PUT_LINE('Sada sam stariji ' || TO_CHAR(  
    ROUND(SYSDATE - TO_DATE('&Dat_rodj',  
    'DD.MM.YYYY'),0)) || ' dana');
```

```
END;
```



# Unos kroz podrazumevani prompt

---

- Dve varijante:
  - *&promenjiva*
    - za svaku promenjivu zahteva se unos vrednosti
  - *&&promenjiva*
    - unos se zahteva samo jednom za vreme trenutne sesija
    - ako je potrebno obrisati vrednost potrebno je izvršiti naredbu
      - `UNDEFINE` *promenjiva*

## Rešenje 2

---

```
UNDEFINE Dat_rodj;
```

```
BEGIN
```

```
  DBMS_OUTPUT.PUT_LINE('Danas je: ' || SYSDATE);
```

```
  DBMS_OUTPUT.PUT_LINE('Rodjen sam: ' || TO_DATE('&&Dat_rodj',  
    'DD.MM.YYYY'));
```

```
  DBMS_OUTPUT.PUT_LINE('Bio je: ' || TO_CHAR(TO_DATE(  
    '&&Dat_rodj', 'DD.MM.YYYY'), 'DAY'));
```

```
  DBMS_OUTPUT.PUT_LINE('Sada sam stariji ' || TO_CHAR(  
    ROUND(SYSDATE - TO_DATE('&&Dat_rodj',  
    'DD.MM.YYYY'),0)) || ' dana');
```

```
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

# Zadatak


---

- Prebrojati projekte. Rezultat smestiti u definisanu varijablu i prikazati ga u konzoli.

# Zadatak

---

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



# Zadatak

---

- Sa tastature uneti broj projekta. Za uneseni broj preuzeti njegove podatke u posebne promenjive i prikazati ih u konzoli.

# Zadatak

---

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;





# Zadatak

---

- U konzolu ispisati radnika (mbr, ime, prezime i platu) čiji je matični broj jednak matičnom broju unetom preko prompta.
- Takođe prikazati jednog radnika čije ime počinje na slovo uneto sa prompta.

# Rešenje

---

```
DECLARE
```

```
    V_MBR RADNIK.MBR%TYPE;
```

```
    V_IME RADNIK.IME%TYPE;
```

```
    V_PRZ RADNIK.PRZ%TYPE;
```

```
    V_PLT RADNIK.PLT%TYPE;
```

```
BEGIN
```

```
    SELECT Mbr, Ime, Prz, Plt
```

```
    INTO  V_MBR, V_IME, V_PRZ, V_PLT
```

```
    FROM  Radnik
```

```
    WHERE mbr = &PR_MBR;
```

```
    DBMS_OUTPUT.PUT_LINE(V_MBR || ' ' || V_IME || ' ' || V_PRZ || ' ' || V_PLT);
```

```
    SELECT Mbr, Ime, Prz, Plt
```

```
    INTO  V_MBR, V_IME, V_PRZ, V_PLT
```

```
    FROM  Radnik
```

```
    WHERE ime like '&PR_IME%' and rownum = 1;
```

```
    DBMS_OUTPUT.PUT_LINE(V_MBR || ' ' || V_IME || ' ' || V_PRZ || ' ' || V_PLT);
```

```
END;
```



# Zadatak

---

- DODATI JOŠ ZADATAKA

# DML naredbe

---

- Normalna upotreba naredbi INSERT, UPDATE i DELETE
- U okviru BEGIN – END sekcije koriste se standardne DML naredbe
  - moguće je zadavati vrednosti preko promenjivih

# 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 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 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;
```

# 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;
```

# Zadatak

---

- Preko tastature uneti podatke o radniku i upisati ga u bazu podataka. Za matični broj uzeti narednu vrednost iz sekvencera.
- Ukoliko je uspešno uneta nova toraka, na konzoli ispisati poruku o uspešnom unosu torke. U protivnom, ispisati poruku o neuspešnom unosu.

# Zadatak

---

```
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 torka u  
tabelu Radnik.');
```

```
    ELSE
```

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

```
    END IF;
```

```
END;
```



# Zadatak

---

- Obrisati radnika čiji matični broj je unet preko tastature. Na konzoli prikazati koliko je radnika obrisano.

# Zadatak

---

```
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 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;



# Izlazak iz petlje / EXIT

---

- EXIT [labela] [WHEN logički\_izraz]
- EXIT se, najčešće, koristi u kombinaciji s bezuslovnom 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;

# 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 deklarirati**. Korak brojača je uvek 1.




# 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;
```



# Zadatak

---

- Ispisati posebno parne i posebno neparne brojeve od 1 do broja unetog sa tastature.

# Zadatak

---

```
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 Prz, Ime, Sef, Plt i God, (za MBR koristiti sekvencer)
- 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 (SEQ\_Mbr.NEXTVAL, '&&Prz', '&&Ime', &&Plt, '&&God');

INSERT INTO radproj (Mbr, Spr, Brc)

VALUES (SEQ\_Mbr.CURRVAL, 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 := &&v_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;
```