PL/SQL feladatok – 8. gyakorlat

1. feladat

Változóhasználat (I. az SQL*Plus felhasználói változói, II. az SQL*Plus környezeti változói, III. a PL/SQL (belső) változói). **Az adatok képernyőre való kiíratása** (Az Oracle beépített csomagjai, DBMS_OUTPUT csomag PUT_LINE eljárása). CONCAT fv. vagy || jel. Írjunk egy PL/SQL blokkot tartalmazó SQL*Plus szkipt programot, amely a felhasználótól bekér egy egész számot! Ha ez a szám nagyobb 100-nál, akkor a PL/SQL blokkban, egyébként pedig a PL/SQL blokkot követő gazdanyelvi (SQL*Plus) környezetben írassa ki.

```
SET serveroutput ON        /*kiiratás engedélyezése*/
ACCEPT valtozo_I PROMPT "Kérem adjon meg egy egész számot: "   /*VALTOZO_I HELYETTESÍTŐ VÁLTOZÓ*/

VARIABLE valtozo_II  NUMBER  /*Gazdakörnyezeti változó deklarálása, VALTOZO_II HOZZÁRENDELT VÁLTOZÓ*/

DECLARE                    /*Változó deklarálása*/
  valtozo_III  NUMBER;
BEGIN
   valtozo_III := &valtozo_I;
   IF valtozo_III > 100
   THEN
      DBMS_OUTPUT.PUT_LINE('A megadott szám: '|| valtozo_III);
   ELSE
     :valtozo_II := valtozo_III;   /*gazdakörnyezeti változónak megadjuk értékül a megadott számot*/
   END IF;
end;

/
PROMPT --> hozzárendelt változó(k) kiiratása a gazdakörnyezetben
      /*Csak akkor lesz a VALTOZO_II alatt érték, ha a megadott szám kisebb, mint 101*/
PRINT valtozo_II

PROMPT --> A hozzárendelt változó(k) lekérdezése a gazdakörnyezetben - variable valtozo_iidatatypes NUMBER
VARIABLE

PROMPT --> GAZDAKÖRNYEZETI és a HELYETTESÍTŐ (- VALTOZO_I) változók lekérdezése
DEFINE


PROMPT --> A helyettesítő változó(k) törlése:
UNDEFINE  valtozo_I

PROMPT --> A gazdakörnyezeti és helyettesítő változók lekérdezése (valtozo_I nem lesz a felsoroltak között):
DEFINE
```

2. feladat

A SELECT ... INTO utasítás a PL/SQL-ben - Határozzuk meg egy PL/SQL program segítségével a felhasználó által megadott telephelyen dolgozók bérösszegét.

```
SET serveroutput ON

ACCEPT részleg PROMPT 'A részleg neve: '

DECLARE
  v_össz_bér   NUMBER;
BEGIN
  SELECT SUM(sal)
    INTO v_össz_bér
    FROM emp, dept
    WHERE UPPER(loc) = UPPER('&részleg')  AND
         emp.deptno = dept.deptno;
  DBMS_OUTPUT.PUT_LINE('A(z) '||'&részleg'||
                    ' részlegen dolgozók havi bére összesen: '||
                    v_össz_bér);
END;
/
```

…

3. feladat

Feltételes utasítás (és a MOD) - Kérjünk be két egész számot és döntsük el, hogy az összegük páros-e vagy páratlan. Az eredményt a PL/SQL blokkban írjuk ki futás közben.

```
SET serveroutput ON
DECLARE
  C NUMBER;
BEGIN
  C := &A +&B;
  DBMS_OUTPUT.PUT_LINE(C);
  IF MOD(C,2) = 1
  THEN
    DBMS_OUTPUT.PUT_LINE(C || '  PÁRATLAN');
  ELSE
    DBMS_OUTPUT.PUT_LINE(C || '  PÁROS');
  END IF;
END;
/
```

4. feladat

LOOP ciklus - Állítsuk elő a felhasználó által megadott darabszámig a Fibonacci-sorozat elemeit. (A megoldáshoz LOOP ciklust használjunk.)

```
SET serveroutput ON
SET echo OFF
SET verify OFF

ACCEPT darabszám PROMPT "Kérem az előállítandó darabszámot: "

DECLARE
  első     NUMBER;
  második  NUMBER;
  új       NUMBER;
  darab    NUMBER;
  számlál  NUMBER;
BEGIN
  darab := &darabszám;
  első  := 0;
  második := 1;
  számlál := 2;
  DBMS_OUTPUT.PUT_LINE('A(z)' || LPAD(1,3) || '. elem:' ||
                    LPAD(első,6));
  DBMS_OUTPUT.PUT_LINE('A(z)' || LPAD(2,3) || '. elem:' ||
                    LPAD(második,6));

  LOOP
    új := első + második;
    számlál := számlál + 1;
    DBMS_OUTPUT.PUT_LINE('A(z)' || LPAD(számlál,3) || '. elem:' ||
                      LPAD(új,6));
    EXIT WHEN számlál = darab;
    első := második;
    második := új;
  END LOOP;
end;
/
```

5. feladat

WHILE ciklus - Kérjünk be két (nem túl nagy) egész számot és írjuk ki a legnagyobb közös osztókat (az Euklideszi algoritmussal és a WHILE ciklust használjunk).

```
SET serveroutput ON
SET echo OFF
SET verify OFF

ACCEPT aa  PROMPT 'Adja meg az első számot:
'
ACCEPT bb  PROMPT 'Adja meg a második
számot: '

VARIABLE lnko  NUMBER

DECLARE
 a        NUMBER;
 b        NUMBER;
 c        NUMBER;
 LépésSzám  NUMBER;
BEGIN
 a := TO_NUMBER(&aa);
 b := TO_NUMBER(&bb);
 LépésSzám := 1;
-- Euklideszi algoritmus:
 WHILE (a<>b)
 LOOP
  IF a<b
  THEN
   c:=a;
   a:=b;
   b:=c;
  END IF;
  a:=a-b;
     DBMS_OUTPUT.PUT_LINE(LépésSzám || '.
lépés: '||a);
  LépésSzám := LépésSzám + 1;
 END LOOP;
 :lnko:=a;
     DBMS_OUTPUT.PUT_LINE('A  legnagyobb
közös osztó: '||a);
END;
/
PROMPT A legnagyobb közös osztó:
PRINT lnko
```

6. feladat

FOR ciklus - Írjunk SQL*Plus szkript programot, amely egy PL/SQL blokkban kiszámítja felhasználó által megadott A számtól a felhasználó által megadott B számig a páratlan számok négyzetösszegét, és ezt az SQL*Plus környezetben írjuk ki. (A megoldáshoz FOR ciklust használjunk.)

```
SET serveroutput ON
SET verify OFF

ACCEPT  Aszám  PROMPT  "Kérem  az  egyik
számot: "
ACCEPT  Bszám  PROMPT  "Kérem  a  másik
számot:  "

VARIABLE négyzetösszeg  NUMBER

DECLARE
 v_Aszám  NUMBER;
 v_Bszám  NUMBER;
 segéd    NUMBER;
 szumma   NUMBER;
BEGIN
 v_Aszám := &Aszám;
 v_Bszám := &Bszám;
 szumma  := 0;
 IF v_Aszám > v_Bszám
 THEN
  segéd   := v_Aszám;
  v_Aszám := v_Bszám;
  v_Bszám := segéd;
 END IF;

 FOR ciklusváltozó IN v_Aszám .. v_Bszám
 LOOP
  IF MOD(ciklusváltozó, 2) != 0
  THEN
   segéd := POWER(ciklusváltozó,2);
            DBMS_OUTPUT.PUT_LINE('A(z)  '||
ciklusváltozó ||
                  ' négyzete: '|| segéd);
   szumma := szumma + segéd;
  END IF;

  :négyzetösszeg := szumma;
 END LOOP;
END;
/

PROMPT  A  megadott  tartomány  páratlan
számainak négyzetösszege:
PRINT négyzetösszeg
```

8. gyakorlat feladatai önálló feldolgozásra

h1a. - Hello World program. Kérjük, hogy adja meg a nevét, és íratassuk ki a képernyőre, hogy 'Szia <név>!' (képernyőre való kiíratás)

h1b. - Írjuk ki KING fizetését! (olvasás táblából változóba)

h1c. - Írjuk ki KING belépési dátumát! (különböző dátum formátumokkal)

h1d. - PL/SQL programmal írassuk ki az emp sorainak számát és az átlagfizetést.

h1e. - Kérjünk be egy (nem túl nagy) egész számot és állapítsuk meg, hogy prím-e.

h1f. - Kérjünk be két egész számot és írjuk ki a legkisebb közös többszörösét.

```
/*--------------------------------------------------------------------------------*/
/*--------------------------------1. FELADAT--------------------------------------*/
/*--------------------------------------------------------------------------------*/

/* h1a. - Hello World program. Kérjük, hogy adja meg a nevét, és íratassuk ki
        a képernyőre, hogy 'Szia <név>!' (képernyőre való kiíratás) */

SET serveroutput ON

ACCEPT be_nev PROMPT "Kérem adja meg a nevét: "

VARIABLE UDVOZLES varchar2(40);

DECLARE
  name varchar2(40) := '&be_nev';
BEGIN
  name := 'Szia ' ||name ||'!';
  DBMS_OUTPUT.PUT_LINE(name);
  :UDVOZLES  := name;
END;
/
PROMPT
PRINT UDVOZLES;

PROMPT  --> A gazdakörnyezeti változók és a helyettesítő változó kiiratása
DEFINE

PROMPT --> Helyettesítő változó törlés (DEFINE BE_NEV...) Kész...
UNDEFINE be_nev

PROMPT  --> Törlés utáni állapot megtekintése
DEFINE

/*--------------------------------------------------------------------------------*/
/*--------------------------------2. FELADAT--------------------------------------*/
/*--------------------------------------------------------------------------------*/

/* h1b. - Írjuk ki KING fizetését! (olvasás táblából változóba).*/

SET serveroutput ON

VARIABLE KING_FIZETESE NUMBER;

DECLARE
 fizetes NUMBER;
BEGIN
  Select sal
   into fizetes
              from emp
                        where  LOWER(ename) = 'king';
   DBMS_OUTPUT.PUT_LINE('King fizetése '||fizetes||' dollár.');
  :KING_FIZETESE := fizetes;
END;

/
PROMPT
PRINT KING_FIZETESE;
```

```
/*--------------------------------------------------------------------------------*/
/*---------------------------------3. FELADAT-------------------------------------*/
/*--------------------------------------------------------------------------------*/
```

/* h1c. - Írjuk ki KING belépési dátumát! (különböző dátum formátumokkal)*/ - 6 - féleképpen megoldva

```
/*--------------------------------------------------------------------------------*/
```

/* Melyik év, melyik hónap, melyik napján lépett be King? */

```
SET serveroutput ON

VARIABLE KING_BELEPESE_ev_honap_nap varchar2(20);

DECLARE
  datum varchar2(20);
BEGIN
   select to_char(hiredate,'YYYY.MM.DD')
          into datum
               from emp
                 where  LOWER(ename) = 'king';
    DBMS_OUTPUT.PUT_LINE('King '||datum||' -án\én lépett be.');
   :KING_BELEPESE_ev_honap_nap := datum;
END;

/
PROMPT
PRINT KING_BELEPESE_ev_honap_nap;
```

/*--*/

/* Melyik év, melyik hónapján lépett be King? */

```
SET serveroutput ON

VARIABLE KING_BELEPESE_ev_honap varchar2(20);

DECLARE
  datum varchar2(20);
BEGIN
   select to_char(hiredate,'YYYY MM.')
          into datum
               from emp
                 where  LOWER(ename) = 'king';
    DBMS_OUTPUT.PUT_LINE('King '||datum||' hónapjában lépett be.');
   :KING_BELEPESE_ev_honap := datum;
END;

/
PROMPT
PRINT KING_BELEPESE_ev_honap;
```

```
/*--------------------------------------------------------------------------------*/

/* Melyik évben lépett be King? */

SET serveroutput ON

VARIABLE KING_BELEPESE_ev NUMBER;

DECLARE
  evszam NUMBER;
BEGIN
   select to_number(to_char(hiredate,'YYYY'))
          into evszam
                from emp
                  where  LOWER(ename) = 'king';
    DBMS_OUTPUT.PUT_LINE('King '||evszam||' -ban/ben lépett be.');
   :KING_BELEPESE_ev := evszam;
END;

/
PROMPT
PRINT KING_BELEPESE_ev;

/*--------------------------------------------------------------------------------*/

/* Hány éve lépett be King? */

SET serveroutput ON

VARIABLE KING_BELEPESE_hany_eve NUMBER;

DECLARE
  evek_szama NUMBER;
BEGIN
   select to_number(to_char(sysdate,'YYYY'))-to_number(to_char(hiredate,'YYYY'))
          into evek_szama
                from emp
                  where  LOWER(ename) = 'king';
    DBMS_OUTPUT.PUT_LINE('King '||evek_szama||' éve lépett be.');
   :KING_BELEPESE_hany_eve := evek_szama;
END;

/
PROMPT
PRINT KING_BELEPESE_hany_eve;
```

```
/*--------------------------------------------------------------------------------*/

/* Hány hónapja lépett be King? */

SET serveroutput ON

VARIABLE KING_BELEPESE_hany_honapja NUMBER;

DECLARE
  honapok_szama NUMBER;
BEGIN
   select round(months_between(sysdate,hiredate))
          into honapok_szama
                from emp
                  where  LOWER(ename) = 'king';
    DBMS_OUTPUT.PUT_LINE('King '||honapok_szama||' hónapja lépett be.');
   :KING_BELEPESE_hany_honapja := honapok_szama;
END;

/
PROMPT
PRINT KING_BELEPESE_hany_honapja;

/*--------------------------------------------------------------------------------*/

/* Hány napja lépett be King? */


SET serveroutput ON

VARIABLE KING_BELEPESE_hany_napja NUMBER;

DECLARE
  napok_szama NUMBER;
BEGIN
   select round(sysdate - hiredate)
          into napok_szama
                from emp
                  where  LOWER(ename) = 'king';
    DBMS_OUTPUT.PUT_LINE('King '||napok_szama||' napja lépett be.');
   :KING_BELEPESE_hany_napja := napok_szama;
END;

/
PROMPT
PRINT KING_BELEPESE_hany_napja;
```

```
/*--------------------------------------------------------------------------------*/
/*-------------------------------4. FELADAT---------------------------------------*/
/*--------------------------------------------------------------------------------*/
```

/* h1d. - PL/SQL programmal írassuk ki az emp sorainak számát és az átlagfizetést. */

```
SET serveroutput ON

VARIABLE Sorok_szama NUMBER;
VARIABLE Atlagfizetes NUMBER;

DECLARE
  sorszam NUMBER;
  atlag NUMBER;
BEGIN
   select count(*)
          into sorszam
                from emp;

   select round(avg(nvl(sal,10000)))
          into atlag
                from emp;

     DBMS_OUTPUT.PUT_LINE('Az emp táblában '||sorszam||' sor van.
                    Az emp táblában tárolt fizetések átlaga: '||atlag||' dollár körülbelül.');
    :Sorok_szama := sorszam;
    :Atlagfizetes := atlag;
END;

/
PROMPT
PRINT Sorok_szama;

PROMPT
PRINT Atlagfizetes;
```

```
/*--------------------------------------------------------------------------------*/
/*-------------------------------5. FELADAT---------------------------------------*/
/*--------------------------------------------------------------------------------*/
```

/*h1e. - Kérjünk be egy (nem túl nagy) egész számot és állapítsuk meg, hogy prím-e. */

```
SET serveroutput ON
SET echo OFF
SET verify OFF

ACCEPT bszam PROMPT 'Kérem adjon meg egy egész számot: '

DECLARE
   szam       NUMBER;
   i          NUMBER;
   d          NUMBER;
   eddig      NUMBER;
   megjegyzes NUMBER;      /*Funkciója: Negatív szám pozitívvá alakítása után a
visszaalakításhoz szükséges. Csak akkor van szükség visszaalakításra, ha az értéke 1*/
```

```
BEGIN
   d := 0;
   megjegyzes := 0;
   szam := TO_NUMBER(&bszam);

   IF szam < 0
      THEN
         szam := szam*(-1);
         megjegyzes := 1;
   END IF;

   IF szam=0
       THEN
          DBMS_OUTPUT.PUT_LINE('A(z) '||szam||' nem prímszám.');
   END IF;

   IF szam=1 or szam=2 or szam=3
       THEN
          DBMS_OUTPUT.PUT_LINE('Az Ön által megadott szám eleme a {-3,-2,-1,1,2,3}
halmaznak. Emiatt a szám prím!');
   END IF;

   IF szam!=0 and szam!=1 and szam!=2 and szam!=3
     THEN
       i := 2;
       eddig := szam - 1;
          WHILE(i<>eddig)
          LOOP
             IF MOD(szam, i) = 0
               THEN
                 d := d + 1;
                 DBMS_OUTPUT.PUT_LINE('A(z) '||i||' osztója '||szam||' -nak ek. Az eddigi
osztók száma: '||d);
               END IF;
               i := i+1;
         END LOOP;

         IF d != 0
           THEN
              IF megjegyzes = 1
                THEN
                   szam := szam*(-1);
              END IF;
             DBMS_OUTPUT.PUT_LINE('A(z) '||szam||' nem prímszám.');
             DBMS_OUTPUT.PUT_LINE('A(z) '||szam||' valós osztóinak száma 1-en és
önmagán kívül: '||d);
           ELSE
              IF megjegyzes = 1
                THEN
                   szam := szam*(-1);
              END IF;
             DBMS_OUTPUT.PUT_LINE('A(z) '||szam||' prímszám.');
         END IF;
   END IF;

END;
/
```

```
/*--------------------------------------------------------------------------------*/
/*--------------------------------6. FELADAT--------------------------------------*/
/*--------------------------------------------------------------------------------*/
```

/* h1f. - Kérjünk be két egész számot és írjuk ki a legkisebb közös többszörösét.*/

```
SET serveroutput ON
SET echo OFF
SET verify OFF

ACCEPT elso_szam PROMPT 'Kérem adja meg az első számot: '
ACCEPT masodik_szam PROMPT 'Kérem adja meg a második számot: '

VARIABLE legkisebb_kozos_tobbszoros NUMBER;

DECLARE
    k           NUMBER;
    a           NUMBER;
    b           NUMBER;
    c           NUMBER;
    d           NUMBER;
    eredeti_1     NUMBER;
    eredeti_2     NUMBER;
    l_k_k_t       NUMBER;

BEGIN
  a := TO_NUMBER(&elso_szam);
  b := TO_NUMBER(&masodik_szam);
  eredeti_1 := a;
  eredeti_2 := b;

   IF a<0
     THEN
        a:=a*(-1); /*A negatív számok miatt kell. Hiánya esetén rossz megoldásra jutnánk*/
  END IF;

  IF b<0
    THEN
      b:=b*(-1); /*A negatív számok miatt kell. Hiánya esetén rossz megoldásra jutnánk*/
  END IF;

  c := a;
  d := b;

  k := 0;
  WHILE (k=0)
   LOOP
     IF a > b
       THEN
         IF MOD(a,b) = 0
            THEN
              :legkisebb_kozos_tobbszoros := a;
              l_k_k_t := a;
              k := 1;
           ELSE a := a+c;
         END IF;
       ELSE
```

```
          IF MOD(b,a) = 0
            THEN
              :legkisebb_kozos_tobbszoros := b;
              l_k_k_t := b;
              k := 1;
            ELSE b := b+d;
        END IF;
      END IF;
    END LOOP;

  DBMS_OUTPUT.PUT_LINE('A(z) '||eredeti_1||' és a(z) '||eredeti_2||' legkisebb közös
többszöröse a(z) '||l_k_k_t||'.');
END;

/
PROMPT --> A felhasználó által megadott két szám legkisebb közös többszöröse:
PRINT legkisebb_kozos_tobbszoros
```

PL/SQL feladatok – 9. gyakorlat

1. feladat
Összetett típus, rekord, gyűjtőtábla - Írjunk PL/SQL programot, amely meghatározza a 7698 azonosítójú dolgozó nevét gyűjtőtábla használatával.

```
SET serveroutput ON
DECLARE
  TYPE  dolg_tabla_tipus IS TABLE OF emp%ROWTYPE
        INDEX BY BINARY_INTEGER;
  dolgozo  dolg_tabla_tipus;
BEGIN
  SELECT *
    INTO dolgozo(1)
    FROM emp
    WHERE empno = 7698;

  IF dolgozo.EXISTS(1)
  THEN
    DBMS_OUTPUT.PUT_LINE(dolgozo(1).ename);
  END IF;
END;
/
```


---

2. feladat
Implicit kurzor, kurzorhasználat FOR ciklusban - Írjunk PL/SQL programot, amely meghatározza a 7698 azonosítójú dolgozó nevét gyűjtőtábla használatával.

```
-- (Rejtett kurzorral)
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS  SELECT * FROM emp;
ALTER TABLE dolgozó
  ADD (sorszám  NUMBER(2));

DECLARE
  v_sorszám  dolgozó.sorszám%TYPE := 1;
BEGIN
  FOR drekord IN (SELECT *
                  FROM dolgozó
                  ORDER BY ename)

  LOOP
    UPDATE dolgozó
      SET sorszám = v_sorszám
      WHERE empno = drekord.empno;
    v_sorszám := v_sorszám + 1;
  END LOOP;
END;
/

SET numwidth 5
SELECT *
  FROM dolgozó
  ORDER BY ename;
SET numwidth 10
```


---

3. feladat - Explicit kurzor - 2b feladat másik megoldása

```
-- (Explicit kurzorral)
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS  SELECT * FROM emp;
ALTER TABLE dolgozó
  ADD (sorszám  NUMBER(2));

DECLARE
   -- Deklarálja a sorszám oszlophoz a v_sorszám
változót,
   -- és ennek kezdőértéke legyen egy
  v_sorszám  dolgozó.sorszám%TYPE := 1;
    -- A kurzor a tábla rendezett sorait tartalmazza
  CURSOR dolg_kurzor IS
    SELECT *
      FROM dolgozó
      ORDER BY ename;
BEGIN
  FOR drekord IN dolg_kurzor
  LOOP
    UPDATE dolgozó
      SET sorszám = v_sorszám
      WHERE empno = drekord.empno;
    v_sorszám := v_sorszám + 1;
  END LOOP;
END;
/

SET numwidth 5
SELECT *
  FROM dolgozó
  ORDER BY ename;
SET numwidth 10
```


---

4. feladat

Explicit kurzor, FOR UPDATE, CURRENT OF
- 2b, 2c feladat harmadik megoldása

```
-- (Explicit kurzorral és CURRENT OF hivatkozással)
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS  SELECT * FROM emp;
ALTER TABLE dolgozó
ADD (sorszám  NUMBER(2));
DECLARE


  v_sorszám  dolgozó.sorszám%TYPE := 1;
  CURSOR dolg_kurzor IS
    SELECT *
      FROM dolgozó
      ORDER BY ename
      FOR UPDATE OF sorszám NOWAIT;

BEGIN
  FOR drekord IN dolg_kurzor
  LOOP
    UPDATE dolgozó
      SET sorszám = v_sorszám
      WHERE CURRENT OF dolg_kurzor;
    v_sorszám := v_sorszám + 1;
  END LOOP;
END;
/

SET numwidth 5
SELECT *
  FROM dolgozó
  ORDER BY ename;
SET numwidth 10
```


---

5. feladat

Kurzor, kurzorattribútumok (%FOUND, %NOTFOUND, stb) - Növeljük meg a hivatalnokok (CLERK) fizetését a saját fizetésük 20%-ával!

```
DROP TABLE dolgozo;

CREATE TABLE dolgozo
AS  SELECT * FROM emp;

SET serveroutput ON

DECLARE
  CURSOR egydolgozo IS
    SELECT empno, ename, sal
      FROM dolgozo
      WHERE UPPER(job) = 'CLERK'
      FOR UPDATE NOWAIT;

  azonosito  dolgozo.empno%TYPE;
  nev        dolgozo.ename%TYPE;
  fizetes    dolgozo.sal%TYPE;


BEGIN
  OPEN egydolgozo;
  LOOP
    FETCH egydolgozo
      INTO azonosito, nev, fizetes;
    EXIT  WHEN egydolgozo%NOTFOUND;
    fizetes := fizetes *1.2;

    UPDATE dolgozo
      SET sal = fizetes
      WHERE CURRENT OF egydolgozo;

    DBMS_OUTPUT.PUT_LINE( nev||'   '||fizetes);
  END LOOP;
  CLOSE egydolgozo;
END;
/

SET numwidth 6
SELECT * FROM dolgozo;
SET numwidth 10
```

6. feladat
hivatkozási és összetett adattípusok - Előző példa, csak tömb (PL/SQL tábla) használatával.

```
-- pelda tomb (pl/sql tabla) hasznalatara

SET SERVEROUTPUT ON

DECLARE
  v_nev VARCHAR2(20);
  CURSOR emp_cur IS
  SELECT deptno, ename FROM emp;
  rec emp_cur%ROWTYPE;
  TYPE tab_tip IS TABLE OF emp_cur%ROWTYPE INDEX BY BINARY_INTEGER;
  tabla tab_tip;
  i  NUMBER(4);
BEGIN
  OPEN emp_cur;
  LOOP
    FETCH emp_cur INTO rec;
    EXIT WHEN emp_cur%NOTFOUND;
    i:= emp_cur%ROWCOUNT;
    tabla(i) := rec;
    dbms_output.put_line(to_char(tabla(i).deptno)||' - '||tabla(i).ename);
  END LOOP;
  CLOSE emp_cur;
END;
/
```

----------Házi feladatok----------
---

h2a. - Írjuk ki a dolgozók nevét és fizetését! (kurzor használata)

```
DROP TABLE munkas;

CREATE TABLE munkas
AS  SELECT * FROM emp;

SET serveroutput ON

DECLARE
  CURSOR egydolgozo IS
    SELECT ename, sal FROM munkas
                    FOR UPDATE NOWAIT;

  nev       munkas.ename%TYPE;
  fizetes    munkas.sal%TYPE;

BEGIN
  OPEN egydolgozo;
    LOOP
      FETCH egydolgozo
      INTO nev, fizetes;
      EXIT  WHEN egydolgozo%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE( nev||'   '||fizetes);
    END LOOP;
  CLOSE egydolgozo;
END;
/
```

```
SET numwidth 6
SELECT ename, sal FROM munkas;
SET numwidth 10
```


---

h2b. - Írjunk PL/SQLprogramot, amely (eldob és) létrehoz az emp táblából egy dolgozo táblát, és megnöveli a felhasználó által megadott foglalkozású dolgozók fizetését 1000 USD-ral.

```
DROP TABLE dolgozo;
CREATE TABLE dolgozo AS SELECT * FROM emp;

SET serveroutput ON

ACCEPT fogl PROMPT "Kérem adja meg, hogy milyen foglalkozású dolgozók fizetését
akarja növelni: "

DECLARE
   foglalkozas dolgozo.job%TYPE := '&fogl';

   CURSOR egydolgozo IS
      SELECT empno, ename, sal
         FROM dolgozo
               WHERE UPPER(job) = UPPER(foglalkozas)
                     FOR UPDATE NOWAIT;

   azonosito dolgozo.empno%TYPE;
   nev         dolgozo.ename%TYPE;
   fizetes      dolgozo.sal%TYPE;

  BEGIN
    OPEN egydolgozo;
      LOOP
         FETCH egydolgozo
           INTO azonosito, nev, fizetes;
      EXIT WHEN egydolgozo%NOTFOUND;
         fizetes := fizetes + 10000;
         UPDATE dolgozo
           SET sal = fizetes WHERE CURRENT OF egydolgozo;

         DBMS_OUTPUT.PUT_LINE(nev||' '||fizetes);
      END LOOP;
    CLOSE egydolgozo;
END;
/

SET numwidth 6
SELECT * FROM dolgozo;
SET numwidth 10
```


---

h2c. - Írjunk PL/SQLprogramot, amely (eldob és) létrehoz az emp táblából egy dolgozo táblát, és megnöveli a felhasználó által megadott százalékértékkel minden, az átlagfizetésnél alacsonyabb fizetéssel rendelkező dolgozók fizetését.

```
DROP TABLE dolgozo;
CREATE TABLE dolgozo AS SELECT * FROM emp;

SELECT * FROM dolgozo;

SET serveroutput ON

ACCEPT szl PROMPT "Kérem adja meg, hogy hány százalékkal növeljem a fizetést:  "

DECLARE
   atlag_fizetes dolgozo.sal%TYPE;
   sz dolgozo.job%TYPE := '&szl';

   CURSOR egydolgozo IS
       SELECT empno, ename, sal
           FROM dolgozo
                WHERE sal < atlag_fizetes
                        FOR UPDATE NOWAIT;

   azonosito dolgozo.empno%TYPE;
   nev         dolgozo.ename%TYPE;
   fizetes      dolgozo.sal%TYPE;

  BEGIN
    sz := sz*0.01;    /*A megadott értéket osztom százzal, hogy százalék értéket kapjak*/
    sz := 1 + sz;    /*S hozzáadok 1et, különben a későbbiek folyamán a fizetés megszorzása
esetében kisebb értéket kapnánk, mint amennyi valójában kellene, hogy legyen*/

   DBMS_OUTPUT.PUT_LINE('NÉV:   |   (Fizetés*'||sz||'):');

    SELECT round(avg(sal))
           INTO atlag_fizetes FROM emp;

    OPEN egydolgozo;
     LOOP
        FETCH egydolgozo
          INTO azonosito, nev, fizetes;
     EXIT WHEN egydolgozo%NOTFOUND;
        fizetes := fizetes * sz;
        UPDATE dolgozo
          SET sal = fizetes WHERE CURRENT OF egydolgozo;

        DBMS_OUTPUT.PUT_LINE(nev||'         |        '||fizetes);
     END LOOP;
    CLOSE egydolgozo;
END;
/

SET numwidth 6
SELECT * FROM dolgozo;
SET numwidth 10
```


---

h2d. - Írjunk PL/SQLprogramot, amely (eldob és) létrehoz az emp táblából egy dolgozo táblát, és ebben foglalkozásonként megnöveli a legkisebb fizetésű dolgozók bérét a foglalkozási csoportjukban legnagyobb fizetés és az ugyanitt számított átlagfizetés különbségének 20%-ával.

```
DROP TABLE dolgozo;
CREATE TABLE dolgozo AS SELECT * FROM emp;

SELECT * FROM dolgozo;

SET serveroutput ON

DECLARE

   CURSOR egydolgozo IS
      SELECT empno, ename, job, sal
         FROM dolgozo
                  FOR UPDATE NOWAIT;

      CURSOR egyfoglalkozas IS                /*Foglalkozások szerint csoportosít, s kiírja a minimális-, maximális-, valamint átlagfizetést foglalkozásonként*/
         SELECT job, min(sal), max(sal), avg(sal)
                  FROM dolgozo GROUP BY job;

   azonosito dolgozo.empno%TYPE;
   nev        dolgozo.ename%TYPE;
   fogl       dolgozo.job%TYPE;
   fizetes     dolgozo.sal%TYPE;

   foglalkozas dolgozo.job%TYPE;
   min_fizetes dolgozo.sal%TYPE;
   max_fizetes dolgozo.sal%TYPE;
   atlag_fizetes dolgozo.sal%TYPE;
   differencia dolgozo.sal%TYPE;
   diff_husz_szazaleka dolgozo.sal%TYPE;
   megnovelt_ber dolgozo.sal%TYPE;

   L      BOOLEAN;

  BEGIN

    OPEN egyfoglalkozas;          /* "egyfoglalkozás" a foglalkozásokat, valamint a foglalkozáson belül a legkisebb, legnagyobb és átlagfizetéseket tárolja. */

      LOOP
         L := TRUE;
         FETCH egyfoglalkozas
           INTO foglalkozas, min_fizetes, max_fizetes, atlag_fizetes;

      EXIT WHEN egyfoglalkozas%NOTFOUND;
```

```
        differencia := atlag_fizetes -  max_fizetes;        /* Minden egyes foglalkozás esetében
kiszámítom a legnagyobb és az átlagfizetés különbségét, */

        IF differencia < 0
            THEN
                differencia := differencia*(-1);
            END IF;

        diff_husz_szazaleka := differencia*1.2;                    /* beszorzom 1.2-del, majd*/
        megnovelt_ber := min_fizetes + diff_husz_szazaleka;       /* ezt az értéket hozzáadom a
legkisebb fizetéshez. Az értéket eltárolom. */

        OPEN egydolgozo;                                          /* Ezek után "egydolgozó"-ban addig
megyek, */
          LOOP
            FETCH egydolgozo
              INTO azonosito, nev, fogl, fizetes;

           EXIT WHEN egydolgozo%NOTFOUND or L=FALSE;              /* amíg meg nem találom
azt a sort, amelyben a foglalkozás neve és a fizetés */

              IF fogl = foglalkozas and fizetes = min_fizetes    /* meg nem egyezik az
"egyfoglalkozás"-ban tárolt foglalkozásnévvel és minimális fizetéssel.*/
              THEN
                fizetes := megnovelt_ber;                  /* Itt megadom értékül a fizetésnek az előbb is
említett eltárolt értéket, */
                L:=FALSE;

                UPDATE dolgozo
                  SET sal = fizetes WHERE CURRENT OF egydolgozo;
                  DBMS_OUTPUT.PUT_LINE(nev||'         |       '||fizetes||'         |       '||
fogl);   /* majd kiiratom a dolgozó nevét, fizetését és a foglalkozásának nevét. */
              END IF;

          END LOOP;
        CLOSE egydolgozo;


     END LOOP;
   CLOSE egyfoglalkozas;
END;
/

SET numwidth 6
SELECT * FROM dolgozo;
SET numwidth 10


/*Utánaszámolás végett, ellenőrzésre:*/

/*select count(ename), job, min(sal), max(sal), avg(sal) from emp group by job;*/

/*MARTIN | 1490 | SALESMAN*/
/*SMITH | 1115 | CLERK*/
/*KING | 5000 | PRESIDENT*/
/*CLARK | 2710 | MANAGER*/
/*FORD | 3000 | ANALYST*/
----------------------------------------------------------------
----------------------------------------------------------------
```

---

h2e. - Írjuk ki a 3. 5. és 8. legnagyobb fizetésű dolgozó nevét, fizetését!
(kurzor attribútumok %ROWCOUNT)

```
DROP TABLE munkas;

CREATE TABLE munkas
AS  SELECT * FROM emp;

SET serveroutput ON

DECLARE
  CURSOR egydolgozo IS
    SELECT ename, sal FROM munkas order by sal
                    FOR UPDATE NOWAIT;

  nev        munkas.ename%TYPE;
  fizetes    munkas.sal%TYPE;

BEGIN
  OPEN egydolgozo;
    LOOP
      FETCH egydolgozo
        INTO nev, fizetes;
      EXIT  WHEN egydolgozo%NOTFOUND;

      IF egydolgozo%ROWCOUNT = 3 or egydolgozo%ROWCOUNT = 5 or egydolgozo
%ROWCOUNT = 8
          THEN
            DBMS_OUTPUT.PUT_LINE(egydolgozo%ROWCOUNT||'. dolgozó neve: '||nev||'.
'||nev||' fizetése: '||fizetes);
      END IF;

    END LOOP;
  CLOSE egydolgozo;
END;
/

SET numwidth 6
SELECT ename, sal FROM munkas;
SET numwidth 10
```


---

h2f. - Tegyük be a dolgozók nevét egy plsql tömbbe, és írjuk ki az utolsó előtti sort!
(összetett adattípusok RECORD/rekord TABLE OF/gyűjtőtábla v. plslq tömb)

```
SET serveroutput ON

DECLARE

 CURSOR nevek IS
       SELECT ename FROM emp;
       rec nevek%ROWTYPE;
  TYPE tablatipus IS TABLE OF
              nevek%ROWTYPE INDEX BY BINARY_INTEGER;

  tabla tablatipus;
  i  NUMBER(10);
  n NUMBER(10);

BEGIN

  OPEN nevek;
  LOOP
    FETCH nevek INTO rec;

     IF  nevek%NOTFOUND
        THEN
          i := n-1;
         dbms_output.put_line(i||'. sorban szereplő név: '||tabla(i).ename||' (Össz sorok száma: '||
n||' )');
     END IF;

    EXIT WHEN nevek%NOTFOUND;
    n:= nevek%ROWCOUNT;      /* n. sornál tartunk */
    tabla(n) := rec;         /* n. sort feltlötjük a tábla n. sorával */

  END LOOP;
  CLOSE nevek;
END;
/
```

PL/SQL feladatok – 10. gyakorlat

1. feladat - Egyszerű számnövelő példa a PL/SQL függvényekre és eljárásokra.

```
-- Nehany egyszeru pelda a pl/sql fuggvenyek
-- es procedurak hasznalatara vonatkozoan

SET SERVEROUTPUT ON


-- Az alabbi blokk alprogramjai nem taroltak, azok csak
-- a blokk utasitasaiban hivhatok
DECLARE
  szam number(6);

  FUNCTION fv_plusz_1(szam number) RETURN number IS
    lokalis_valtozo NUMBER(6);   /*Visszadjuk az értéket*/
  BEGIN
    lokalis_valtozo := szam + 1;
    return(lokalis_valtozo);     /*Visszatérési érték*/
  END;

  PROCEDURE pr_plusz_1(szam number) is
    lokalis_valtozo NUMBER(6);
  BEGIN
    lokalis_valtozo := szam + 1;
    dbms_output.put_line(TO_CHAR(lokalis_valtozo));
  END;


BEGIN
  szam := fv_plusz_1(100);

  pr_plusz_1(szam);
END;
/
-- Az alabbi alprogramok viszont taroltak, azok az adatbazisban
-- tarolodnak es a kesobbiekben barmikor hivhatok.
-- A fv SQL utasitasban is hasznalhato (a procedura csak PL/SQL-ben).

CREATE OR REPLACE FUNCTION fv_plusz_2(szam number) RETURN number IS
  lokalis_valtozo NUMBER(6);
BEGIN
  lokalis_valtozo := szam + 2;
  return(lokalis_valtozo);
END;
/

SELECT fv_plusz_2(1000) FROM dual;


CREATE OR REPLACE PROCEDURE pr_plusz_2(szam number) is
  lokalis_valtozo NUMBER(6);
BEGIN
  lokalis_valtozo := szam + 2;
  dbms_output.put_line(TO_CHAR(lokalis_valtozo));
END;
/

BEGIN
  pr_plusz_2(2000);
END;
/
-- Vagy a fentivel ekvivalens meghivasi mod SQLPLUS-bol

EXECUTE pr_plusz_2(2000);
```

---

2. feladat – csomag - Egyszerű példa egy getemp eljárást és avg_salary fv-t tartalmazó csomagra.

```
-- pelda csomagra

CREATE PACKAGE emp_pkg IS
  PROCEDURE getemp;
  FUNCTION avg_salary RETURN NUMBER;
END emp_pkg;
/

CREATE PACKAGE BODY emp_pkg IS        /*Csomag törzse, Az egyes eljárások, függvények leírása,
implementálása*/

  PROCEDURE getemp IS  -- header
    emp_id  employees.employee_id%type;
    lname   employees.last_name%type;
  BEGIN
    emp_id := 100;
    SELECT last_name INTO lname
    FROM hr.EMPLOYEES
    WHERE employee_id = emp_id;
    DBMS_OUTPUT.PUT_LINE('Last name: '||lname);
  END;


  FUNCTION avg_salary RETURN NUMBER IS
    avg_sal employees.salary%type;
  BEGIN
    SELECT AVG(salary) INTO avg_sal
    FROM EMPLOYEES;
    RETURN avg_sal;
  END;

END emp_pkg;
/
```


---

3. feladat - hiba és kivételkezelés - Példa hiba és kivételkezelésre (módosítsuk úgy a programot, hogy másik ágra tereljük a hibakezelést...)

```
-- pelda hiba es kivetelkezelesre
-- ha apro modositasokat irunk a programba, vagy commentbe
-- teszunk sorokat, masik agra terelhetjuk a hibakezelest

DECLARE
  v_nev  VARCHAR2(20);
  v_szam NUMBER := 0;
  CURSOR emp_cur IS SELECT ename FROM emp;

  hiba1 EXCEPTION;
  pragma EXCEPTION_INIT(hiba1, -20001);
  hiba2 EXCEPTION;
  pragma EXCEPTION_INIT(hiba2, -20002);

  PROCEDURE hibas_proc(szam NUMBER) IS
  BEGIN
    IF MOD(szam, 2) = 0 THEN
      RAISE_APPLICATION_ERROR('-20001', 'Elso hiba');
    ELSE
      RAISE_APPLICATION_ERROR('-20002', 'Masodik hiba');
    END IF;
  END;
```

```
BEGIN
  hibas_proc(1);
  v_szam := 1/v_szam;  -- nullaval osztas
  SELECT ename INTO v_nev FROM emp WHERE empno < 0;  -- no_data/too_many_rows
  OPEN emp_cur;
  LOOP
    FETCH emp_cur INTO v_nev;
    EXIT WHEN emp_cur%notfound;
    dbms_output.put_line(v_nev);
    dbms_output.put_line(to_char(emp_cur%rowcount));
  END LOOP;
  CLOSE emp_cur;
EXCEPTION
  WHEN hiba1 THEN
    dbms_output.put_line('Elso hiba fordult elo');
  WHEN hiba2 THEN
    dbms_output.put_line('Masodik hiba fordult elo');
  WHEN zero_divide THEN
    dbms_output.put_line('Nullaval osztas hiba');
  WHEN no_data_found THEN
    dbms_output.put_line('No Data Found hiba');
  WHEN too_many_rows THEN
    dbms_output.put_line('Too many rows hiba');
  WHEN OTHERS THEN
    dbms_output.put_line('Valami mas jellegu hiba ...');
END;
/
```

---

4. feladat

- kivételkezelés

- Írjunk egy olyan eljárást, amely kivételkezelést is tartalmaz és a jutalmat az emp táblából létrehozott dolgozo tábla jutalék (comm) értékéhez adja hozzá! A jutalom a dolgozó fizetésének 10%-a, feltéve, ha a fizetés 3000 dollár alatt van, egyébként csak egy "Boldog Karácsonyt!" üdvözletet kap.

```
SET serveroutput ON
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS  SELECT * FROM emp;

-- JutalmazóProgram
ACCEPT neve  PROMPT 'Adja meg a jutalmazandó dolgozó nevét: '
-- A főprogram deklarációs szegmense
DECLARE
  v_neve       dolgozó.ename%TYPE;
  v_jutalma    dolgozó.sal%TYPE;
  NincsJutalom  EXCEPTION;

-- Az alprogram deklaráció
PROCEDURE Jutalmazás (p_neve     IN   dolgozó.ename%TYPE,
                      p_jutalma  OUT  dolgozó.sal%TYPE)
IS
-- Lokális deklaráció
  p_fizetése  dolgozó.sal%TYPE;
BEGIN
  SELECT sal
    INTO p_fizetése
    FROM dolgozó
    WHERE UPPER(ename) = UPPER(p_neve);
  IF p_fizetése >= 3000
  THEN
    RAISE NincsJutalom;
  ELSE
    p_jutalma := 0.1 * p_fizetése;
    DBMS_OUTPUT.PUT_LINE('>>');
    DBMS_OUTPUT.PUT_LINE('>> ' || p_neve ||' jutalma: '||
```

```
                        p_jutalma ||' USD');
END IF;

EXCEPTION
  WHEN NO_DATA_FOUND
    THEN
      DBMS_OUTPUT.PUT_LINE('>>');
      DBMS_OUTPUT.PUT_LINE('>> NINCS ilyen nevű dolgozó...');
  WHEN NincsJutalom
    THEN
      DBMS_OUTPUT.PUT_LINE('>>');
      DBMS_OUTPUT.PUT_LINE('>> Boldog Karácsonyt Kedves ' ||
                    p_neve || '!');
END Jutalmazás;
-- Az alprogram deklarációjának vége

-- Főprogram blokkja
BEGIN
  v_neve := '&neve';
  Jutalmazás(v_neve, v_jutalma);
  UPDATE dolgozó
    SET comm = NVL(comm,0) + v_jutalma
    WHERE UPPER(ename) = UPPER(v_neve);
END; -- JutalmazóProgram
/

-- PROMPT Listázás:
SELECT ename       AS "Név",
       job         AS "Munkakör",
       sal         AS "Fizetés",
       NVL(comm,0)  AS "Jutalék+Jutalom"
  FROM dolgozó
  WHERE UPPER(ename) = UPPER('&neve');
```

/*Házi feladatok*/

/* 1. feladat (h3a.) */
/* Írjunk meg egy függvényt, ami az empno azonosító alapján visszaadja a nevet! */

/* ---------- */

```
DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

ACCEPT azon PROMPT 'Kérem, adja meg a dolgozó azonosítóját: '

Set serveroutput ON

DECLARE
  d_azon dolgozó.empno%TYPE;
  dolg_nev dolgozó.ename%TYPE;

FUNCTION azonosit(d_azon dolgozó.empno%TYPE) RETURN dolgozó.ename%TYPE IS
  d_nev dolgozó.ename%TYPE;
BEGIN
  SELECT ename INTO d_nev
       FROM dolgozó WHERE dolgozó.empno = d_azon;
  return(d_nev);

EXCEPTION
   WHEN NO_DATA_FOUND
     THEN
       DBMS_OUTPUT.PUT_LINE('Nincs ilyen azonosítójú dolgozó!');
```

```
END azonosit;

BEGIN
  d_azon := &azon;
  dolg_nev := azonosit(d_azon);
  dbms_output.put_line('A(z) ' ||d_azon||' azonosítójú hallgató neve: '||
TO_CHAR(dolg_nev));
END;
/

Select empno AS "ID",
        ename AS "Név"
        FROM dolgozó WHERE empno = &azon;

/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */

/* 2. feladat (h3b.) */
/* Írjunk egy olyan függvényt, amely kurzor technikával egy EmpnoIn változóval */
/* megegyező dolgozó nevét betölti egy EnameOut változóba! */

/* ---------------------------------------------------------------------- */

DROP TABLE dolgozó;
CREATE TABLE dolgozó
AS SELECT * FROM emp;

ACCEPT Empno_IN PROMPT 'Kérem, adja meg a dolgozó nevét: '

Set serveroutput ON

DECLARE
  EmpnoIN dolgozó.empno%TYPE;
  EnameOUT dolgozó.ename%TYPE;

  CURSOR egysor IS
       SELECT empno, ename from dolgozó
                    FOR UPDATE NOWAIT;

  CURSOR dolgozok_szama IS
       SELECT count(ename) FROM dolgozó;

FUNCTION betolt(EmpnoIN dolgozó.empno%TYPE) RETURN dolgozó.ename%TYPE
    IS l BOOLEAN;
       azon dolgozó.empno%TYPE;
       EnameOUT dolgozó.ename%TYPE;
       dolgozokszama number;
       Szamlalo number;
BEGIN
   Szamlalo := 0;
   OPEN dolgozok_szama;
   LOOP
     FETCH dolgozok_szama
          INTO dolgozokszama;
     EXIT WHEN dolgozok_szama%NOTFOUND;
   END LOOP;
   CLOSE dolgozok_szama;
```

```
  OPEN egysor;
    l := TRUE;
    LOOP
      FETCH egysor
       INTO azon, EnameOUT;        /*Ha nem létező azonosítót adok meg, akkor is eltárolja
azonosítókat, neveket */
       Szamlalo := Szamlalo + 1;
    EXIT WHEN egysor%NOTFOUND or l=FALSE;

       IF EmpnoIN = azon     /* Ha nem létező azonosítót adok meg, akkor ebbe az elágazásba
be sem lép a program, hanem */
         THEN                /* az összes soron végig megy, s végül névnek az utolsó dolgozó
nevét felelteti meg */
           l := FALSE;
       END IF;

       IF EmpnoIN != azon and Szamlalo = dolgozokszama  /* Ha az azonosító az utolsó
esetben sem egyezik meg, akkor a név */
         THEN                          /* Egy szóközzel lesz egyenlő. Ez a főprogramban
"kihasználható"...*/
           EnameOUT := ' ';
       END IF;
    END LOOP;
  CLOSE egysor;

  return(EnameOut);

EXCEPTION
   WHEN no_data_found THEN
   dbms_output.put_line('Nincs ilyen azonosítójú dolgozó');
   WHEN OTHERS THEN
   dbms_output.put_line('Valamilyen mas jellegu hiba!');

END betolt;

BEGIN
  EmpnoIN := &Empno_IN;
  EnameOUT := betolt(EmpnoIN);
   IF EnameOUT = ' '
      THEN
         dbms_output.put_line('Nincs ilyen azonosítójú dolgozó');
      ELSE
        dbms_output.put_line('A(z) ' ||EmpnoIN||' azonosítójú hallgató neve: '||
TO_CHAR(EnameOUT));
   END IF;
END;
/

Select empno AS "ID",
       ename AS "Név"
       FROM dolgozó WHERE empno = &Empno_IN;

/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */
```

/* 3. feladat (h3c.) */
/* Módosítsuk a fizetéseket egy kurzorral végighaladva rajtuk! Adjunk hozzá mindenki fizetéséhez*/
/* n*10 ezret, ahol n a nevében levő magánhangzók (vagyis a, e, i, o, u) száma! */

/* ---------- */

/* Először is lássunk egy példaprogramot, ami segítségével egy adott string betűit vizsgálhatjuk meg: */

```
Set serveroutput ON;

DECLARE
  nev varchar2(40);
  hossz number;
  maradek varchar2(40);
  i number;

BEGIN
  nev := 'Eötvös Lóránd TudományEgyetem';
  hossz := Length(nev);
  i:=hossz-1;
  WHILE i>=0
  LOOP
    maradek := SUBSTR(nev,hossz-i,1);
    dbms_output.put_line(maradek);
    i:=i-1;
  END LOOP;
END;
/
```

/* Most lássuk az eredeti problémát és oldjuk meg a feladatot: */

```
DROP TABLE dolgozo;

CREATE TABLE dolgozo
AS  SELECT * FROM emp;

SET serveroutput ON

DECLARE
 mhz_szama NUMBER;

 CURSOR adottsor IS
   SELECT empno, ename, sal
     FROM dolgozo
       FOR UPDATE NOWAIT;

 azonosito  dolgozo.empno%TYPE;
 nev        dolgozo.ename%TYPE;
 fizetes    dolgozo.sal%TYPE;

FUNCTION szamolok_maganhangzokat(dolg_nev dolgozo.ename%TYPE) RETURN
NUMBER IS
  hossz NUMBER;
  akt_betu VARCHAR2(40);
  mhz_szama NUMBER;
```

```
  i NUMBER;
BEGIN
  mhz_szama := 0;
  hossz := Length(nev);
  i:=hossz-1;

  WHILE i>=0
   LOOP
     akt_betu := SUBSTR(nev,hossz-i,1);

     IF akt_betu = 'a' or akt_betu = 'e' or akt_betu = 'i' or akt_betu = 'o' or akt_betu = 'u'
     THEN
       mhz_szama := mhz_szama + 1;
     END IF;

     i:=i-1;
   END LOOP;
   return(mhz_szama);
END;

BEGIN
 OPEN adottsor;
 LOOP
   FETCH adottsor
     INTO azonosito, nev, fizetes;
   EXIT  WHEN adottsor%NOTFOUND;

   nev := LOWER(nev);
   mhz_szama :=  szamolok_maganhangzokat(nev);

   DBMS_OUTPUT.PUT_LINE( nev||' nevében '||mhz_szama||' db magánhangzó van,
fizetése ');
   DBMS_OUTPUT.PUT_LINE(mhz_szama*10000||' USD-vel fog növekedni.');
   DBMS_OUTPUT.PUT_LINE(nev||' eddigi fizetése '||fizetes||' USD volt.');

   fizetes := fizetes + mhz_szama*10000;

   UPDATE dolgozo
     SET sal = fizetes
     WHERE CURRENT OF adottsor;   /*Mindig az aktuális sort írja felül.*/

  DBMS_OUTPUT.PUT_LINE(' Előzőek értelmében '||nev||' fizetése '||fizetes||' USD lett.');
  DBMS_OUTPUT.PUT_LINE('-------');
 END LOOP;
 CLOSE adottsor;
END;
/

SET numwidth 6
SELECT * FROM dolgozo;
SET numwidth 10

/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */
```

```
/* 4. feladat (h3d.) */
/* Írjunk egy csomagot, amelyben eljárásként, függvényként, tárolt eljárásként      */
/* és tárolt függvényként is szerepel az, hogy adott n-re visszaadja n faktoriálist! */
/* Ha a faktoriális túl nagy szám lenne, akkor a függvény adjon vissza -1-et,       */
/* hiba és kivételkezeléssel megoldva, ekkor egy üzenetet is írjunk ki a hibáról.   */
/* ---------------------------------------------------------------------- */

DROP PACKAGE CSOMAG;

CREATE PACKAGE csomag IS
     PROCEDURE proc_fakt(sz IN NUMBER, eredm OUT NUMBER);

     FUNCTION fugv_fakt(szam NUMBER) RETURN NUMBER;

     CREATE OR REPLACE PROCEDURE tarolt_proc_fakt(sz IN NUMBER, eredm OUT
NUMBER) ;

     CREATE OR REPLACE FUNCTION  tarolt_fugv_fakt(szam NUMBER) RETURN
NUMBER;
END csomag;
/

CREATE PACKAGE BODY csomag IS

/* ELJÁRÁS */

PROCEDURE proc_fakt(sz IN NUMBER, eredm OUT NUMBER) IS
   i NUMBER;
   j NUMBER;
BEGIN

  i := 1;
  eredm := sz;
   WHILE i<sz
   LOOP
     j := sz-i;
     eredm := eredm * j;
     i := i+1;
   END LOOP;

END;

/* FÜGGVÉNY */

FUNCTION  fugv_fakt(szam NUMBER) RETURN NUMBER IS
   eredmeny NUMBER;
BEGIN

  i := 1;
  eredm := sz;
   WHILE i<sz
   LOOP
     j := sz-i;
     eredm := eredm * j;
     i := i+1;
   END LOOP;

   return(eredmeny);
```

```
END;
```

/* TÁROLT ELJÁRÁS */

```
CREATE OR REPLACE PROCEDURE tarolt_proc_fakt(sz IN NUMBER, eredm OUT
NUMBER) IS
  i NUMBER;
  j NUMBER;
BEGIN

 i := 1;
 eredm := sz;
  WHILE i<sz
  LOOP
    j := sz-i;
    eredm := eredm * j;
    i := i+1;
  END LOOP;

END;
```

/* TÁROLT FÜGGVÉNY */

```
CREATE OR REPLACE FUNCTION tarolt_fugv_fakt(szam NUMBER) RETURN
NUMBER IS
  eredmeny NUMBER;
BEGIN

 i := 1;
 eredm := sz;
  WHILE i<sz
  LOOP
    j := sz-i;
    eredm := eredm * j;
    i := i+1;
  END LOOP;

  IF sz > 83
  THEN
    eredm := -1;
  END IF;

  return(eredmeny);

END;

END csomag;
/

SET serveroutput ON;

ACCEPT szam_be PROMPT 'Kérem adjon meg egy nem túl nagy, 0-nál nagyobb egész
számot: '

DECLARE
  szam     NUMBER;
  eredmeny NUMBER;
```

```
  negatív_érték EXCEPTION;
  túlcsordulás EXCEPTION;

BEGIN
  szam := &szam_be;

  IF szam < 0
    THEN
      RAISE negatív_érték;
  END IF;

  IF szam > 83
    THEN
      RAISE túlcsordulás;
  END IF;

  EXCEPTION
  WHEN no_data_found
   THEN
     dbms_output.put_line('Nem adott meg értéket!');
  WHEN negatív_érték
   THEN
     dbms_output.put_line('Negtaív számot nem adhat meg!');
  WHEN túlcsordulás
   THEN
     dbms_output.put_line('Túl nagy értéket adott meg!');

  proc_fakt(szam, eredmeny);    /* eljárás */

  IF szam > 83
    THEN
      eredmeny := -1;
  END IF;

  dbms_output.put_line('Eljárásként kiszámolva: '||szam||'! = '||eredmeny);
  eredmeny := fugv_fakt(szam);    /* függvény */

  IF szam > 83
    THEN
      eredmeny := -1;
  END IF;

  dbms_output.put_line('Függényként kiszámolva: '||szam||'! = '||eredmeny);
  tarolt_proc_fakt(szam,eredmeny);    /* tárolt eljárás */

  IF szam > 83
    THEN
      eredmeny := -1;
  END IF;

  dbms_output.put_line('Tárolt eljárásként kiszámolva: '||szam||'! = '||eredmeny);
  eredmeny := tarolt_fugv_fakt(szam);    /* tárolt függvény */

  IF szam > 83
    THEN
      eredmeny := -1;
  END IF;
```

```
  dbms_output.put_line('Tárolt függényként kiszámolva: '||szam||'! = '||eredmeny);
END;
/

/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */
```

/* 5. feladat Fibonacci */
/* A Fibonacci sorozat megvalósítása fügvénnyel*/
/* -- */

```
SET serveroutput ON;

ACCEPT meddig PROMPT 'Kérem adja meg, hogy a Fibonacci-sorozat hány elemét adjam
meg: '

DECLARE
  eddig NUMBER;
  fib NUMBER;
  negatív_érték EXCEPTION;
  túlcsordulás EXCEPTION;

FUNCTION fibonacci(szam NUMBER, eddig NUMBER) RETURN NUMBER IS
    akt     NUMBER;
    i       NUMBER;
    akt_el  NUMBER;
    akt_ell NUMBER;
BEGIN
   IF eddig > 605
     THEN
       RAISE túlcsordulás;
   END IF;

   IF eddig < 0
     THEN
       RAISE negatív_érték;
   END IF;

   i := 1;
   akt_ell := szam;
   dbms_output.put_line('A Fibonacci-sorozat '||i||'. eleme a(z) '||akt_ell||'. :)');
   i := i + 1;
   akt_el := akt_ell + 1;
   dbms_output.put_line('A Fibonacci-sorozat '||i||'. eleme a(z) '||akt_el||'. :)');
   WHILE i<eddig
     LOOP
      i := i + 1;
      akt:= akt_ell + akt_el;
      dbms_output.put_line('A Fibonacci-sorozat '||i||'. eleme a(z) '||akt||'. :)');
      akt_ell := akt_el;
      akt_el := akt;
    END LOOP;
   return(akt);

EXCEPTION
  WHEN no_data_found
    THEN
```

```
      dbms_output.put_line('Nem adott meg értéket!');
  WHEN negatív_érték
    THEN
      dbms_output.put_line('Negatív számot nem adhat meg!');
  WHEN túlcsordulás
    THEN
      dbms_output.put_line('Túl nagy számot adott meg!');

END;

BEGIN
    eddig := &meddig;
    fib := fibonacci(0,eddig);
END;
/
```

PL/SQL feladatok – 11. gyakorlat

1. feladat

Hozzunk létre BEFORE triggert, amely megakadályozza a munkaidőn kívüli adatmanipulációkat az emp táblán! Írassuk ki, milyen műveleteket kíséreltek meg végrehajtani munkaidőn kívül!

```
CREATE OR REPLACE TRIGGER NeNyúljHozzá
BEFORE DELETE OR INSERT OR UPDATE ON emp
BEGIN
  IF TO_CHAR(sysdate, 'HH24:MI') NOT BETWEEN '08:00' AND '14:30'
  THEN
    IF DELETING THEN
      RAISE_APPLICATION_ERROR(-20211,
        'Csak munkaidőben szabad adatot törölni!');
    ELSIF INSERTING THEN
      RAISE_APPLICATION_ERROR(-20212,
        'Csak munkaidőben szabad adatot bevinni!');
    ELSE
      RAISE_APPLICATION_ERROR(-20213,
        'Csak munkaidőben szabad adatot módosítani!');
    END IF;
  END IF;
END;
/

DELETE FROM emp WHERE upper(ename) = 'CLERK';
INSERT INTO emp
  VALUES (1234,'KISS','president',NULL,sysdate,6000,NULL,10);
UPDATE emp SET ename = 'CLARK' WHERE upper(ename) = 'CLERK';

SHOW ERRORS

-- Ellenőrzés
-- DELETE FROM emp WHERE UPPER(ename) = UPPER('Smith');
```


---

2. feladat

Írjunk triggert (és ellenőrizzük is a működését), amely megakadályozza az elnökre (president) vonatkozó törlő, beszúró és adatmódosító DML utasítások működését! (a tesztelő script programban a hivatkozási megszorítás felfüggesztése illetve a végén az újbóli engedélyeztetése)

```
CREATE OR REPLACE TRIGGER KivéveAzElnök
BEFORE DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
WHEN (UPPER(OLD.job) = 'PRESIDENT'  OR
     UPPER(NEW.job) = 'PRESIDENT')
BEGIN
  IF DELETING THEN
    RAISE_APPLICATION_ERROR(-20001,'>> Az elnököt nem lehet törölni...');
  ELSIF INSERTING THEN
    RAISE_APPLICATION_ERROR(-20002,'>> Elnököt nem lehet beszúrni...');
  ELSIF UPDATING THEN
    RAISE_APPLICATION_ERROR(-20003,'>> Elnök adatok nem módosíthatók...');
  END IF;
END;
/
SHOW ERRORS


-- Ellenőrzés
```

```
-- A tesztelő szkript program eleje
SET serveroutput ON
PROMPT Hivatkozási megszorítás felfüggesztése:
ALTER TABLE emp
  DISABLE CONSTRAINT EMP_SELF_KEY;

UPDATE emp
  SET sal = sal - 555
  WHERE deptno = 10;

INSERT INTO emp
  VALUES (1234,'KISS','president',NULL,sysdate,6000,NULL,10);

DELETE FROM emp
  WHERE deptno = 10;

PROMPT Hivatkozási megszorítás engedélyezése:
ALTER TABLE emp
  ENABLE CONSTRAINT EMP_SELF_KEY;
SET numwidth 5
SELECT * FROM emp;
SET numwidth 10
-- A tesztelő szkript program vége

-- tesztelés:

select * from emp order by  deptno;

UPDATE emp
  SET sal = sal - 555
  WHERE deptno = 10;

select * from emp order by deptno;
```


---

3. feladat

Adj meg egy olyan triggert, amely törli a hallgató összes adatát az indexk táblából miután töröltük a hallgatót a hallgato táblából!

```
/*********************************************************/
/* Sémát, lásd hátul, megszorítások nélkül hozzuk lére, normálisan a hallgato tábla ehakod  oszlopának PRIMARY KEY
megszorítást adnánk, és az indexk tábla ehakod oszlopa FOREIGN KEY idegen kulcs lenne az alábbi hivatkozási integritási
megszorítással: REFERENCES hallgato (ehakod) ON DELETE CASCADE. Ez többek között azt is biztosítja, hogy a
hallgato szülőtábla egy sorának törlésekor a rendszer lépcsőzetesen az indexk gyermektábla függő sorait is törölje. */
/*********************************************************/
/*  trigger létrehozása  */

CREATE OR REPLACE TRIGGER HallgatoKaszkadTorles
        AFTER DELETE
                ON hallgato
                FOR EACH ROW
BEGIN
        IF DELETING THEN
                DELETE FROM indexk
                        WHERE ehakod = :OLD.ehakod;
END IF;
END;
/
SHOW ERRORS;

/*  Ellenőrzés: jól működik-e a trigger? */
SAVEPOINT A;                        /* Kimentjük a tartalmat */
PROMPT Az eredeti táblák listázása:
SELECT * FROM hallgato ORDER BY ehakod;
```

```
SELECT * FROM indexk ORDER BY ehakod;
PROMPT Törlési próba:
DELETE FROM hallgato
WHERE UPPER(CIM) = 'BUDAPEST';
PROMPT A módosított táblák listázása:
SELECT * FROM hallgato ORDER BY ehakod;
SELECT * FROM indexk ORDER BY ehakod;

/*  Takarítás: Adatvisszaállítás és a trigger törlése */
ROLLBACK TO A;
DROP TRIGGER HallgatoKaszkadTorles;
PROMPT Újra az eredeti táblák listázása:
SELECT * FROM hallgato ORDER BY ehakod;
SELECT * FROM indexk ORDER BY ehakod;

/*********************************************************/
/*   Táblák (a 3a miatt megszorítások nélkül való) létrehozása és pár adat bevitele: */

drop table indexk cascade constraints;
drop table hallgato cascade constraints;

create table hallgato
      (ehakod varchar2(35),
       nev varchar2(35),
       cim varchar2(35));

create table indexk
      (ehakod varchar2(35),
       targy varchar2(35),
       jegy number(5),
       datum date);

insert into hallgato values ('1', 'Péter', 'Budapest');
insert into hallgato values ('2', 'Pál', 'Eger');
insert into hallgato values ('3', 'András', 'Budapest');
insert into hallgato values ('4', 'Jakab', 'Debrecen');
insert into hallgato values ('5', 'János', 'Pécs');
insert into hallgato values ('6', 'Máté', 'Szeged');
insert into hallgato values ('7', 'Márk', 'Budapest');

insert into indexk values ('1', 'adatbázis1', 4, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('1', 'adatbázis2', 1, SYSDATE);
insert into indexk values ('1', 'hálózatok', 1, TO_DATE('2007.05.21','YYYY.MM.DD'));
insert into indexk values ('2', 'adatbázis1', 1, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('2', 'adatbázis2', 1, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('2', 'hálózatok', 3, SYSDATE);
insert into indexk values ('3', 'adatbázis1', 5, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('3', 'adatbázis2', 4, SYSDATE);
insert into indexk values ('3', 'hálózatok', 5, SYSDATE);
insert into indexk values ('4', 'adatbázis1', 5, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('4', 'adatbázis2', 1, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('4', 'hálózatok', 1, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('5', 'adatbázis1', 5, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('5', 'adatbázis2', 5, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('5', 'hálózatok', 5, TO_DATE('2007.06.27','YYYY.MM.DD'));
insert into indexk values ('6', 'hálózatok', 2, SYSDATE);

commit;

select * from hallgato order by ehakod;
select * from indexk order by ehakod;

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```