8. FEJEZET

Változóhasználat, vezérlési szerkezetek a PL/SQL-ben

Elméleti összefoglaló

A következő három fejezet az Oracle magasszintű, professzionális alkalmazások létrehozására szolgáló programozási nyelvét, a PL/SQL-t mutatja be a nyelvi alapoktól az összetett és hivatkozási típusok használatán keresztül a kurzorok, tárolt eljárások és triggerek témaköréig. Különös jelentőséget ad a PL/SQL nyelvnek az a tény, hogy a legújabb szabványbővítések éppen a PL/SQL-eszközök beépítésével viszik procedurális, illetve objektumorientált irányba a hagyományosan deklaratív SQL nyelvet, vagyis úgy tűnik, hogy a nem is távoli jövő a teljes alkalmazásfejlesztésre képes SQL nyelvé [20].

A PL/SQL az Oracle által kifejlesztett magasszintű programnyelv, a hagyományos SQL nyelv procedurális kiterjesztése. Megtalálhatók benne a korszerű programozási nyelvek jellemzői, így többek közt a kivételkezelés és az objektumhasználat. A PL/SQL programok írásával tehát egyesíthetjük a deklaratív SQL nyelv tömörségét egy procedurális nyelv algoritmusleírásra, teljes alkalmazások kifejlesztésre való képességével.

Az Oracle a PL/SQL programokat külön „motorral", a PL/SQL fordítóprogrammal dolgozza fel, és ez független az SQL-utasításokat feldolgozó „SQL-motortól". Ennek hatására növekszik a teljesítmény. A „külön motor" kiszűri az SQL-utasításokat és továbbítja az SQL-motor felé. A PL/SQL a szerveroldalon fut, a fordítóprogram része a PL/SQL programokat egy köztes (pszeudo) kódba fordítja, így futtatáskor nem kell lexikális elemzést végezni, és a futtató programja futás előtt még optimalizálja is ezt a kódot. (A PL/SQL nyelv részletesebb ismertetését *lásd* [15].)

PL/SQL blokk

Az SQL*Plus-környezetben a PL/SQL nyelvű programozás alapfogalma a (névtelen) blokk, ahol a blokk leginkább a hagyományos program fogalmának felel meg. Egy PL/SQL blokk sokszor csak egy egyszerű utasítássorozat, amelyet esetleg egy SELECT utasítás vesz körül valamilyen interaktív környezettel annak érdekében, hogy a felhasználó azt az aktuális igényeinek megfelelően tudja paraméterezni. Más esetben azonban egy összetett vezérlési szerkezet, amely egy bonyolult táblaalgoritmust valósít meg. A különböző igényekhez különböző eszközöket biztosít a PL/SQL.

Egy PL/SQL blokkot az SQL*Plus- (mint gazdanyelvi) környezetben egy SQL*Plus szkript programba ágyazva futtatunk. A beágyazás módja a következő: a bevezető (általában adatbekérést is tartalmazó) SQL*Plus- és SQL-utasítások után következik a PL/SQL blokk, melyet a / jel zár le. Ezt további SQL*Plus- és SQL-utasítások, valamint PL/SQL blokkok követhetnek.

Egy blokk három, funkcionálisan jól elkülöníthető részből, úgynevezett szegmensből állhat: a *deklarációs*, a *végrehajtható* és a *kivételkezelő* részből. A deklarációs rész és a kivételkezelő rész használata opcionális, míg a végrehajtható rész használata kötelező.

Deklarációs szegmens

A deklarációs szegmenst a `DECLARE` kulcsszó vezeti be. Ez tartalmazza a blokkban felhasznált összes változót, kurzort és a felhasználó által definiált kivételeket.

Végrehajtható szegmens

A `BEGIN` és az `EXCEPTION`, vagy annak hiányában az `END` kulcsszavak között találhatók SQL-utasítások, PL/SQL-utasítások, alprogramhívások, vezérlési szerkezetek. A végrehajtható szegmensbe újabb PL/SQL blokkok ágyazhatók be.

Megjegyezzük, hogy az SQL-utasítások egy része csak módosított szintaktikával használható a PL/SQL blokkban.

Kivételkezelő szegmens

Az `EXCEPTION` kulcsszó vezeti be. A végrehajtandó részben felmerülő hibák, és abnormális állapotok esetén végrehajtandó utasításokat határozza meg. Helye az `END` utasítás előtt van. (Részletesen *lásd* 10. fejezet.)

A blokkok szerkezete

A blokk felépítésének általános alakja:

```
[DECLARE
   deklarációk]
BEGIN
   végrehajtható utasítások
[EXCEPTION
   kivételkezelés]
END;
/
```

Megjegyzés
Sose felejtkezzünk el a PL/SQL blokkot lezáró / jel használatáról!

Változók használata

Az SQL*Plus-környezetben történő PL/SQL programozás során háromféle változót használhatunk:

- az SQL*Plus felhasználói (helyettesítő) változóit,
- az SQL*Plus környezeti (gazdakörnyezeti) változóit,
- a PL/SQL (belső) változóit.

Az alábbiakban röviden áttekintjük ezeket, illetve egymással való kapcsolatukat.

I. Az SQL*Plus felhasználói változói

Egy PL/SQL blokk részére az őt futtató SQL*Plus-környezetből adatot bevinni a (4. fejezetben már részletesen ismertetett) felhasználói (más néven helyettesítő) változókkal lehetséges. (E változók értékének beolvasása a PL/SQL blokkon belül *&változó* módon történhet). Értéküket megőrzik a PL/SQL blokkból való visszatérés után is.

II. Az SQL*Plus környezeti változói

Egy PL/SQL blokkból adatokat kivinni az őt futtató környezetükbe az úgynevezett környezeti (gazdakörnyezeti vagy hozzárendelt) változókkal lehetséges. E változók az SQL*Plus-környezetben a

```
VARIABLE  változónév  adattípus
```

utasítás segítségével hozhatók létre, ahol az *adattípus* a következők valamelyike lehet:

`NUMBER`	esetleges negatív előjelet és/vagy tizedespontot tartalmazó szám,
`CHAR[(n)]`	kötött hosszúságú karaktersorozat, vagyis a definícióban megadottnál rövidebb karaktersorozatokat balra tömöríti és jobbról szóköz karakterekkel tölti fel, az *n* maximális mérete 2000, alapértelmezett értéke 1,
`VARCHAR2(n)`	változó hosszúságú karaktersorozat, vagyis a definícióban megadottnál rövidebb karaktersorozatokat a tényleges hosszúságuk szerint kezeli (például jeleníti meg), az *n* maximális mérete 4000.

E változók a PL/SQL blokkban kaphatnak értéket értékadó utasítással. (Ott a PL/SQL belső változóitól való megkülönböztetés érdekében a nevük elé egy kettőspont (`:`) karaktert kell tenni.) A környezeti változó az SQL*Plus-környezetből való kilépésig őrzi meg az értékét (akkor törlődik). Az Oracle által javasolt karakter típus a `VARCHAR2(n)`.

Környezeti változó tulajdonságainak lekérdezése:

```
VARIABLE [változónév]
```

*Környezeti változó értékének lekérdezése (csak az az SQL*Plus-környezetben lehetséges):*

```
PRINT [változónév]
```

III. A PL/SQL változói

A PL/SQL nyelv igen sokféle változó használatát biztosítja. A PL/SQL-változókat (az úgynevezett belső változókat) deklarációs utasítással a PL/SQL blokkban hozzuk létre, és az abból való visszalépéskor automatikusan megsemmisülnek. (A változók létrehozására, hatáskörére és az alkalmazható adattípusokra vonatkozó részletes ismertetést *lásd* [15], [16] és [26].)

A belső változók deklarációja a deklarációs szegmensben történik a `DECLARE` kulcsszót követően:

```
DECLARE
  változónév   adattípus;
  [változónév   adattípus;]
  ...
```

ahol az egyes változók deklációjában (az elemi deklarációkban) szereplő adattípusok lehetnek *egyszerű* és *összetett típusok*.

A PL/SQL-ben használható egyszerű típusok egyrészt az 5. fejezetben a *Tábla létrehozása* c. pontban bemutatott `NUMBER[(m [, t] )]`, `CHAR[(n)]`, `VARCHAR2(n)` és `DATE` típusok, másrészt a további szám, illetve karakteres típusok (például `INTEGER`, `REAL`, `FLOAT`, `BINARY_INTEGER`, illetve `STRING`, `LONG` stb.), a nagy méretű (például multimédiás) adatok tárolására szolgáló típusok (`LOB`, `BLOB`, `CLOB`) és a logikai `BOOLEAN` típus (mely a szokásos `TRUE` és `FALSE` értéken kívül felveheti a `NULL` értéket is).

A PL/SQL összetett adattípusaival (hivatkozási típusok, rekord, gyűjtőtábla, kurzor stb.) a következő fejezetben foglalkozunk.

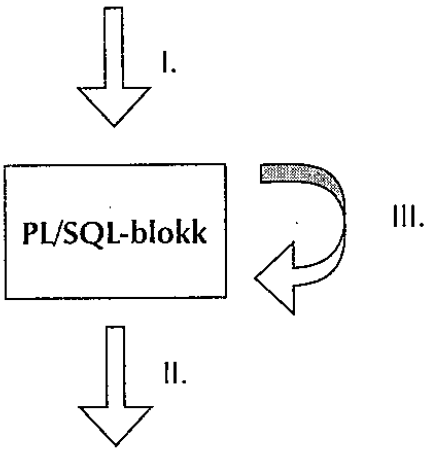
A változóhasználat szemléltetése

SQL*Plus-szkript
I. Értékadással létrehozzuk és lekérdezhetjük
II. Bevezetjük

> **PL/SQL blokk**
> I. Lekérdezhetjük
> II. Értéket kaphat, értéket adhat
> III. Bevezetjük, értéket kaphat, értéket adhat

SQL*Plus szkript (folytatás)
I. Lekérdezhető
II. Lekérdezhető
III. Nem elérhető

Vázlatosan:



8.1. példa

Írjon PL/SQL blokkot tartalmazó SQL*Plus szkript programot, mely 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.

Megoldás

```
-- A DBMS_OUTPUT.PUT_LINE eljárással történő
-- PL/SQL kiíratás engedélyezése:
SET serveroutput ON
-- Felhasználó által megadott változó definiálása
-- és értékének bekérése:
ACCEPT valtozo_I PROMPT "Kérem adjon meg egy egész számot: "
-- Hozzárendelt (gazdakörnyezeti) változó deklarálása:
VARIABLE valtozo_II  NUMBER
-- A PL/SQL-blokk deklarációs szegmense:
-- Változó deklarálása:
DECLARE
  valtozo_III  NUMBER;
-- A PL/SQL-blokk végrehajtási szegmense:
BEGIN
-- Értékadás:
  valtozo_III := &valtozo_I;
-- Feltételes utasítás:
  IF valtozo_III > 100
  THEN
    -- PL/SQL futás közbeni kiíratás
    DBMS_OUTPUT.PUT_LINE('A megadott szám: '|| valtozo_III);
  ELSE
    -- Hozzárendelt változó felveszi a PL/SQL-változó értékét
   :valtozo_II := valtozo_III;
  END IF;
```

```
-- A PL/SQL-blokk vége
END;
-- A PL/SQL-blokk futtatása:
/

PROMPT --> A hozzárendelt változó kiíratása a gazdakörnyezetben:
PRINT  valtozo_II

PROMPT --> A hozzárendelt változók lekérdezése a gazdakörnyezetben:
-- (Ezeket nem kell törölni,
--  mivel az SQL*Plus-ból való kilépéskor törlődnek)
VARIABLE

PROMPT --> A gazdakörnyezeti és helyettesítő változók lekérdezése:
-- (Ezek is törlődnek az SQL*Plus-ból való kilépéskor,
--  de ezek törölhetők is UNDEFINE utasítással)
DEFINE

PROMPT --> A helyettesítő változók törlése:
UNDEFINE  valtozo_I

PROMPT --> A gazdakörnyezeti és helyettesítő változók lekérdezése:
DEFINE
```

1. futtatás után

```
Kérem adjon meg egy egész számot: 555
régi   6:   valtozo_III := &valtozo_I;
új   6:   valtozo_III := 555;
A megadott szám: 555

A PL/SQL eljárás sikeresen befejeződött.

--> A hozzárendelt változó kiíratása a gazdakörnyezetben:

VALTOZO_II
----------


--> A hozzárendelt változók lekérdezése a gazdakörnyezetben:
változó   valtozo_ii
adattípus   NUMBER
--> A gazdakörnyezeti és helyettesítő változók lekérdezése:
DEFINE _CONNECT_IDENTIFIER = "myora" (CHAR)
DEFINE _SQLPLUS_RELEASE = "902000100" (CHAR)
```

```
DEFINE _EDITOR          = "Notepad" (CHAR)
DEFINE _O_VERSION       = "Personal Oracle9i Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production" (CHAR)
DEFINE _O_RELEASE       = "902000100" (CHAR)
DEFINE _RC              = "1" (CHAR)
DEFINE VALTOZO_I        = "555" (CHAR)
--> A helyettesítő változók törlése:
--> A gazdakörnyezeti és helyettesítő változók lekérdezése:
DEFINE _CONNECT_IDENTIFIER = "myora" (CHAR)
DEFINE _SQLPLUS_RELEASE = "902000100" (CHAR)
DEFINE _EDITOR          = "Notepad" (CHAR)
DEFINE _O_VERSION       = "Personal Oracle9i Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production" (CHAR)
DEFINE _O_RELEASE       = "902000100" (CHAR)
DEFINE _RC              = "1" (CHAR)
```

2. futtatás után

```
Kérem adjon meg egy egész számot: 33
régi   6:   valtozo_III := &valtozo_I;
új   6:   valtozo_III := 33;

A PL/SQL eljárás sikeresen befejeződött.

--> A hozzárendelt változó kiíratása a gazdakörnyezetben:

VALTOZO_II
----------
        33

--> A hozzárendelt változók lekérdezése a gazdakörnyezetben:
változó   valtozo_ii
adattípus   NUMBER
--> A gazdakörnyezeti és helyettesítő változók lekérdezése:
DEFINE _CONNECT_IDENTIFIER = "myora" (CHAR)
DEFINE _SQLPLUS_RELEASE = "902000100" (CHAR)
DEFINE _EDITOR          = "Notepad" (CHAR)
DEFINE _O_VERSION       = "Personal Oracle9i Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production" (CHAR)
DEFINE _O_RELEASE       = "902000100" (CHAR)
DEFINE _RC              = "1" (CHAR)
DEFINE VALTOZO_I        = "33" (CHAR)
```

```
--> A helyettesítő változók törlése:
--> A gazdakörnyezeti és helyettesítő változók lekérdezése:
DEFINE _CONNECT_IDENTIFIER = "myora" (CHAR)
DEFINE _SQLPLUS_RELEASE = "902000100" (CHAR)
DEFINE _EDITOR          = "Notepad" (CHAR)
DEFINE _O_VERSION       = "Personal Oracle9i Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production" (CHAR)
DEFINE _O_RELEASE       = "902000100" (CHAR)
DEFINE _RC              = "1" (CHAR)
```

Vezérlési szerkezetek, utasítások

A SELECT utasítás a PL/SQL-ben

Egy adattábla egyes sorait PL/SQL-ben a SELECT utasítás alábbi alakjával kérdezhetjük le (v. ö. az 1. fejezetben bemutatott alakkal):

```
SELECT SzelekciósLista
  INTO {ElemiVáltozó [, ElemiVáltozó]... | RekordVáltozó}
  FROM TáblaLista
  [WHERE LogikaiOszlopkifejezés]
  [GROUP BY CsoportosítóOszlopkifejezés-lista]
  [HAVING LogikaiOszlopkifejezés]
  [ORDER BY RendezőOszlopkifejezés-lista];
```

ahol az elemi és a rekordváltozók a PL/SQL-ben deklarált változók. A PL/SQL-ben a SELECT utasítás WHERE logikai kifejezését úgy kell megadni, hogy a lekérdezés egyszerre csak egy sort adhasson vissza. A szelekciós listában csak azok az oszlopok, és azok is csak olyan sorrendben szerepelhetnek, melyekhez rendelt elemi változók az INTO kulcsszó után állnak, minthogy sorrend szerinti összerendeléssel veszik fel az oszlopértékeket. (Értelemszerűen hasonló követelmény áll fenn a rekord típusú változó adatmezőivel kapcsolatban is.)

Feltételes utasítás

Az utasítás általános alakja:

```
IF feltétel THEN
      utasítások
[ELSIF feltétel THEN
      utasítások]
...
[ELSE
      utasítások]
END IF;
```

ahol:

- *feltétel* logikai változó vagy kifejezés, TRUE, FALSE és NULL értéke lehet,
- *utasítások* egy vagy több PL/SQL- vagy SQL-utasítás. Ezek is tartalmazhatnak további feltételes utasításokat,
- THEN a *feltétel* teljesülése esetén a végrehajtás ága,
- ELSIF ha az első feltétel hamis, akkor a vezérlés ezekre az ágakra adódik, ahol további feltételek adhatók,
- ELSE ha egyetlen feltétel sem teljesül, akkor a vezérlés erre az ágra adódik,
- END IF a feltételes utasítás lezárása.

Ciklusutasítások

A LOOP-CIKLUS

A LOOP-ciklus (vagy más néven egyszerű ciklus) a ciklusmag (a LOOP és az END LOOP közötti rész) utasításait feltétel nélkül hajtja végre. Kilépés az EXIT vagy az EXIT WHEN utasításokkal történhet. (E kilépő utasításrészek a ciklusmag tetszőleges helyén állhatnak, akár egy feltételes utasítás valamelyik ágán is.)

```
LOOP
  utasítás;
  [utasítás;]
  ...
  [EXIT;]
  [EXIT WHEN feltétel;]
END LOOP;
```

A FOR-CIKLUS

A FOR-ciklus ciklusmagja egy LOOP-ciklust tartalmaz, és ez a ciklusmag meghatározott számú feldolgozására használatos. A ciklusváltozója (melynek hatóköre maga a ciklus, azon kívül nincs is értelmezve) sorra felveszi az adathalmazának értékeit, miközben minden ciklusváltozó értékhez végrehajtja a ciklusmag utasításait. A FOR-ciklus rendezett adathalmazát alapértelmezésben a legkisebb és legnagyobb értékek megadásával jelöljük ki. E halmaznak ekkor egyesével növekvő egész számokból kell állnia. (A következő fejezetben látni fogjuk, hogy a FOR-ciklus általános esetben tetszőleges rendezett halmazon is működhet.) Az alapértelmezett FOR-ciklusban a rendezett adathalmaz elemeinek bejárása egyesével növekvő, illetve a REVERSE kulcsszó használata esetén egyesével csökkenő.

```
FOR ciklusváltozó IN [REVERSE] alsóhatár .. felsőhatár
LOOP-ciklus
```

A ciklusváltozó deklarációja automatikus (nem deklarálható külön), és csak a cikluson belül van értéke. Az index név nem lehet ciklusváltozó, mert ez az Oracle-ben kulcsszó. Megjegyezzük,

hogy a későbbiekben a FOR-ciklusnak bemutatjuk azt az alakját, ahol az IN kulcsszó után egy rendezett halmaz (explicit vagy rejtett kurzor) is állhat (*lásd* 9. fejezet).

A WHILE-CIKLUS

A WHILE-ciklus egy elöltesztelő ciklus. A ciklusmagjában lévő LOOP-ciklust addig hajtja végre, amíg a ciklusfejben lévő feltétel teljesül.

```
WHILE feltétel
LOOP-ciklus
```

Megjegyzés
Az EXIT utasítás használata szempontjából az egyszerű LOOP-ciklus és a FOR-, illetve WHILE-ciklusok között az a lényeges különbség, hogy míg a LOOP-ciklusból csak ennek segítségével lehet kijutni (különben végtelen ciklussá válna), addig a másik két ciklus esetén ez csak kiegészítő lehetőség.

Adatok kiírása képernyőre

Az úgynevezett DBMS-csomagok (melyek az objektumorientált programozás objektumaihoz hasonlítanak) a PL/SQL-alkalmazások fejlesztését segítik. A specifikációs részük egy interfészt biztosít, amelyen keresztül hozzáférhetünk a csomag eszközeihez, a törzsrészük pedig tartalmazza a futtatható kódot (részletesen *lásd* [15]).

Az Oracle beépített csomagjai közé tartozik a DBMS_OUTPUT csomag, mely adatok képernyőre való kiíratását végzi. Egy PL/SQL programon belül egy (konstansokból, változókból, függvényekből és műveletekből álló) kifejezés képernyőre történő kiírása e csomag PUT_LINE eljárásával (metódusával) lehetséges a

```
DBMS_OUTPUT.PUT_LINE[(kifejezés)];
```

utasítás segítségével, ahol a *kifejezés* egy karaktersorozat típusú kifejezés, mely tetszőleges SQL karakteres függvényt tartalmazhat. (Ha a *kifejezés* hiányzik, akkor a képernyőn egy soremelés történik.) Ez az eljárás automatikus (karaktersorozattá történő) konverziót végez az adatokon a kiíratás érdekében. Több elem kiíratása esetén a szokásos || jelet, vagy a CONCAT függvényt használhatjuk. Ciklusból (jellemzően listaszerűen, egymás alá) történő kiíratáskor az oszlopszerű formátum érdekében célszerű az RPAD és az LPAD függvényeket alkalmazni.

A DBMS_OUTPUT csomag képernyőre való kiíratását az SQL*Plus-ban engedélyeztetni kell, mely a

```
SET serveroutput ON
```

utasítással tehető meg.

8.2. példa

Kérjen be két egész számot és döntse el, hogy az összegük páros-e vagy páratlan. Az eredményt futás közben (a PL/SQL blokkban) írja ki.

1. megoldás

```
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;
/
```

Eredmény

```
Adja meg a(z) a értékét: 34
Adja meg a(z) b értékét: 23
régi   4:   C := &A +&B;
új   4:   C := 34 +23;
57
57  PÁRATLAN

A PL/SQL eljárás sikeresen befejeződött.
```

2. megoldás

```
SET serveroutput ON
ACCEPT A NUMBER PROMPT'Egyik: '
ACCEPT B PROMPT 'Másik: '

DECLARE
  C   NUMBER;
  CA  NUMBER;
  CB  NUMBER;
BEGIN
  CA := &A;
  CB := &B;
  C  := CA + CB;
```

```
  IF MOD(C,2) = 1
  THEN
    DBMS_OUTPUT.PUT_LINE(CA||'+'||CB||'='||C|| ',  PÁRATLAN');
  ELSE
    DBMS_OUTPUT.PUT_LINE(CA||'+'||CB||'='||C|| ',  PÁROS');
  END IF;
END;
/
```

Eredmény

```
Egyik: 222
Másik: 444
régi   6:   CA := &A;
új   6:   CA :=        222;
régi   7:   CB := &B;
új   7:   CB := 444;
222+444=666,  PÁROS

A PL/SQL eljárás sikeresen befejeződött.
```

3. megoldás

```
SET serveroutput ON

VARIABLE C        NUMBER
VARIABLE PARITAS  VARCHAR2(10)

ACCEPT A NUMBER PROMPT 'Egyik: '
ACCEPT B PROMPT 'Másik: '

BEGIN
  :C := &A + TO_NUMBER(&B);

  IF MOD(:C,2) = 1
  THEN
    :PARITAS :='PÁRATLAN';
  ELSE
    :PARITAS := 'PÁROS';
  END IF;
  DBMS_OUTPUT.PUT_LINE('Az összeg: '||:C||',  '|| :PARITAS);
END;
/
```

Eredmény

```
Egyik: 555
Másik: 111
régi   2:    :C := &A + TO_NUMBER(&B);
új   2:    :C :=        555 + TO_NUMBER(111);
Az összeg: 666,  PÁROS

A PL/SQL eljárás sikeresen befejeződött.
```

8.3. példa

Írjon PL/SQL programot, amely kiírja az emp tábla sorainak számát és az átlagfizetést.

Megoldás

```
SET serveroutput ON
VARIABLE sorok_száma  NUMBER
VARIABLE átlag        NUMBER

DECLARE
  v_létszám   NUMBER;
  v_átlag     NUMBER(7,2);
BEGIN
  SELECT COUNT(*), ROUND(AVG(sal))
    INTO v_létszám, v_átlag
    FROM emp;
  DBMS_OUTPUT.PUT_LINE ('létszám: '||v_létszám||
                        '   átlagfizetés: '||v_átlag);
  :sorok_száma := v_létszám;
  :átlag := v_átlag;
END;
/

PRINT sorok_száma
PRINT átlag
```

Eredmény

```
létszám: 14   átlagfizetés: 2073

A PL/SQL eljárás sikeresen befejeződött.

SOROK_SZÁMA
-----------
         14
```

```
      ÁTLAG
----------
      2073
```

Véletlen számok generálása

Az Oracle beépített csomagjai közé tartozik a DBMS_RANDOM csomag is, mely pszeudovéletlen számokat állít elő az Oracle véletlenszám-generátorából.

A véletlenszám-generátor függvény alakja

```
DBMS_RANDOM.VALUE(AlsóHatár, FelsőHatár)
```

ahol az *AlsóHatár* és a *FelsőHatár* egyaránt Binary_Integer típusú számok.

8.4. példa

Állítsunk elő egy 0 és 10 közötti véletlen számot.

Megoldás

```
SELECT DBMS_RANDOM.VALUE(0,10)
  FROM dual;
```

Eredmény

```
DBMS_RANDOM.VALUE(0,10)
-----------------------
             2.06535198
```

8.5. példa

Állítsunk elő egy 50 és 150 közötti véletlen számot.

Megoldás

```
SELECT DBMS_RANDOM.VALUE(50,150)
  FROM dual;
```

1. futási eredmény

```
DBMS_RANDOM.VALUE(50,150)
-------------------------
               107.463214
```

2. futási eredmény

```
DBMS_RANDOM.VALUE(50,150)
-------------------------
               138.737076
```

8.6. példa

Állítsunk elő egy -50 és +50 közötti véletlen számot.

Megoldás

```
SELECT DBMS_RANDOM.VALUE(-50,+50)
  FROM dual;
```

Eredmény

```
DBMS_RANDOM.VALUE(-50,+50)
--------------------------
                -23.502705
```

8.7. példa

Karácsonyi nyereményjátékot tartanak az emp tábla vállalatánál. Minden dolgozó 0 és 100 USD közötti jutalmat nyerhet, melyet mindjárt hozzá is adnak a jutalékához. Készítse el a játék programját.

1. megoldás

```
SELECT azonosító, név, fizetés, jutalék, nyeremény,
       NVL(jutalék,0)+nyeremény  AS "ÚjJutalék"
  FROM (SELECT empno  AS azonosító,
               ename  AS név,
               sal    AS fizetés,
               comm   AS jutalék,
               ROUND(DBMS_RANDOM.VALUE(0,100))  AS nyeremény
          FROM emp);
```

2. megoldás

```
SELECT empno  AS azonosító,
       ename  AS név,
       sal    AS fizetés,
       comm   AS jutalék,
       nyeremény,
       NVL(comm,0)+ nyeremény  AS "ÚjJutalék"
  FROM emp,
       (SELECT ROUND(DBMS_RANDOM.VALUE(0,100))  AS nyeremény
          FROM dual);
```

Eredmény (mindkét esetben hasonló)

```
AZONOSÍTÓ NÉV        FIZETÉS    JUTALÉK  NYEREMÉNY  ÚjJutalék
--------- ---------- ---------- -------- ---------- ----------
     7839 KING             5000                   4          4
```