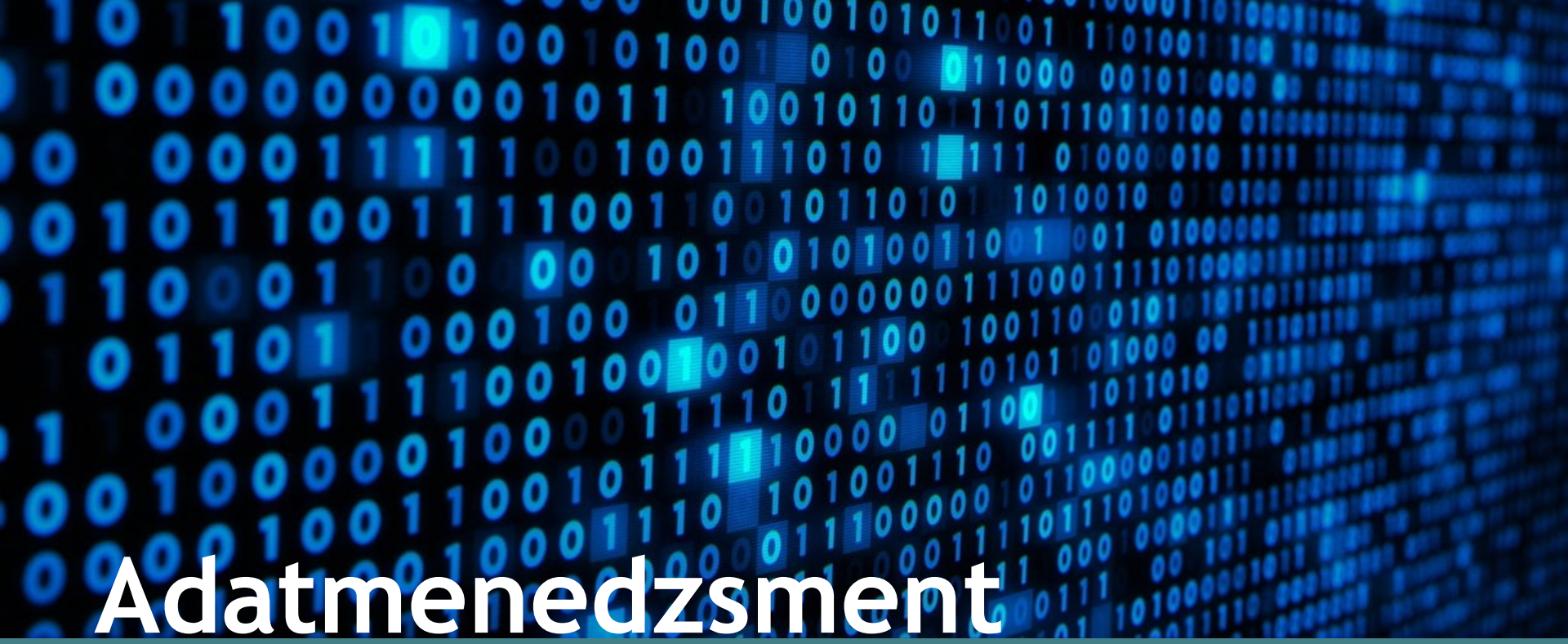

Adatmenedzsment

2. gyakorlat

PL/SQL

- PL/SQL: Procedural Language for SQL
  - Az SQL adatkezelési lehetőségeit kombinálja egy procedurális nyelv lehetőségeivel.
  - Blokkszerkezetű nyelv; szintaxisa Ada-szerű.
- A PL/SQL a következő konstrukciókkal egészíti ki az SQL-t:
  - változók és típusok; vezérlési szerkezetek; alprogramok és csomagok; kurzorok és kurzorváltozók; kivételkezelés; objektumorientált eszközök

A PL/SQL blokk

- A blokk a PL/SQL alapvető programegysége.
  - Kezelhető önállóan és beágyazható más programegységbe.
  - A blokk megjelenhet bárhol a programban, ahol végrehajtható utasítás állhat.
- A blokknak három alapvető része van:
  - deklarációs rész *(opcionális)*,
  - végrehajtható rész
  - kivételkezelő rész *(opcionális)*

A PL/SQL blokk

```
[címke]
[DECLARE
  deklarációs utasítás(ok)]
BEGIN
  végrehajtható utasítás(ok)
  [EXCEPTION
    kivételkezelő utasítás(ok)]
END [név];
```

A PL/SQL blokk

- Címke:
  - A PL/SQL-programban bármely végrehajtható utasítás címkézhető.
  - A címke egy azonosító, amelyet a << és >> szimbólumok határolnak, és az utasítás előtt áll.
- Hatáskör:
  - Egy lokális név a deklarációjától kezdve látszik.
  - A deklarációk hatásköre a blokk végéig tart.

A PL/SQL blokk - példák

- 1. példa

```
BEGIN
   null;
END;
```

- 2. példa

```
<<pelda>>
BEGIN
   DELETE FROM EMPLOYEES;
END;
```

A deklarációs rész

- Deklarációs rész tartalmazhat:
  - típus definíciót
  - változó deklarációt
  - nevesített konstans deklarációt
  - kivétel deklarációt
  - kurzor definíciót
  - alprogram definíciót
  - (pragmát)

A deklarációs rész

- A deklaráció:
  - tárhelyet foglal le a megadott adattípusú értékhez és
  - elnevezi a tárhelyet, hogy hivatkozni tudjuk.
- Fontos: az objektumokat mindig deklarálni kell, mielőtt hivatkozzuk őket.
- Deklaráció szerepelhet blokkok, alprogramok, és csomagok deklarációs részében is.

Változók és nevesített konstansok

- Változó

  ```
  név típus [[NOT NULL] {:= | DEFAULT} kifejezés];
  ```

  Például:

  ```
  szuletesi_ido DATE;
  dolgozok_szama NUMBER NOT NULL DEFAULT 0;
  ```

- Nevesített konstans

  ```
  név CONSTANT típus [NOT NULL] {:= | DEFAULT}
  kifejezés;
  ```

  Például:

  ```
  max_napok_szama CONSTANT NUMBER := 366;
  ```

Változók és nevesített konstansok

- A név egy azonosító, azaz egy olyan karaktersorozat, amely betűvel kezdődik és esetlegesen betűvel, számjeggyel, illetve a $, _, # karakterekkel folytatódhat.
- Egy azonosító maximális hossza 30 karakter és minden karakter szignifikáns.
- A PL/SQL a kis- és nagybetűket nem tekinti különbözőnek!

Változók és nevesített konstansok

- Ha egy változónak nem adunk explicit kezdőértéket, akkor alapértelmezett kezdőértéke NULL lesz.
- A nevesített konstansnál kötelező a kifejezés megadása.
- A nevesített konstansok és változók kezdőértékadása mindig megtörténik valahányszor aktivizálódik az a blokk vagy alprogram, amelyben deklaráltuk őket.
- A skaláris változókra és konstansokra alkalmazhatunk NOT NULL megszorítást.
- A NOT NULL megszorítással rendelkező elemhez kezdőértéket kell rendelni.

Típusok

- Az eddig tanult SQL típusok:
  - VARCHAR2
  - CHAR
  - NUMBER
  - DATE
  - …

  (Később kiegészítésre kerülnek a PL/SQL saját típusaival.)

A DBMS_OUTPUT csomag

- A PL/SQL nem tartalmaz I/O utasításokat.
  - A DBMS_OUTPUT csomag PUT_LINE eljárásának segítségével üzeneteket helyezhetünk el egy belső pufferben.
- A puffer tartalmának megjelenítése:
  - SET SERVEROUTPUT ON
- Példa:

```
BEGIN
   DBMS_OUTPUT.PUT_LINE('Hello, World!');
END;
```

Utasítások

- Üres utasítás: NULL;
- Értékadó utasítás: x := 10;
- Ugró utasítás: GOTO címke;
- Elágaztató utasítások:
  - Feltételes utasítás
  - CASE utasítás
- Ciklusok
- SQL utasítások

Utasítások

- Üres utasítás: NULL;
  - Átadja a vezérlést a következő utasításnak
  - Hol hasznos?
    - Fejlesztés
    - Kivételkezelés
- Értékadó utasítás: x := 10;
  - Az értékadó utasítás egy változó, mező, kollekcióelem vagy objektum attribútum értékét állítja be. Az értéket mindig egy kifejezés segítségével adjuk meg.

Feltételes elágaztató utasítás

```
IF feltétel
   THEN utasítás [utasítás]
   [ELSIF feltétel
     THEN utasítás [utasítás]…]…
   [ELSE utasítás [utasítás]…]
END IF;
```

Feltételes elágaztató utasítás

- Példa:

```
DECLARE
   v_nagyobb NUMBER;
   x NUMBER := 7;
   y NUMBER := 6;
BEGIN
   v_nagyobb := x;
   IF x < y
     THEN v_nagyobb := y;
   END IF;
END;
```

Feltételes elágaztató utasítás

- Példa:

```
DECLARE
  v_nagyobb NUMBER; x NUMBER := 7; y NUMBER := 6;
BEGIN
  IF x < y
    THEN v_nagyobb := y;
    ELSE v_nagyobb := x;
  END IF;
END;
```

Feltételes elágaztató utasítás

- Példa:

```
DECLARE
  v_nagyobb NUMBER; x NUMBER := 5; y NUMBER := 7;
BEGIN
  IF x < y THEN
    DBMS_OUTPUT.PUT_LINE('x kisebb, mint y');
    v_nagyobb := y;
  ELSIF x > y THEN
    DBMS_OUTPUT.PUT_LINE('x nagyobb, mint y');
    v_nagyobb := x;
  ELSE
    DBMS_OUTPUT.PUT_LINE('x és y egyenlők');
    v_nagyobb := x;
  END IF;
END;
```

Feltételes elágaztató utasítás

- Példa:

```
DECLARE
  v_nagyobb NUMBER; x NUMBER := 5; y NUMBER := 7; z NUMBER := 6;
BEGIN
  IF x > y
    THEN IF x > z
           THEN v_nagyobb := x;
           ELSE v_nagyobb := z;
         END IF;
    ELSE IF y > z
           THEN v_nagyobb := y;
           ELSE v_nagyobb := z;
         END IF;
  END IF;
END;
```

CASE elágaztató utasítás

- Általános alak:

```
CASE [szelektor_kifejezés]
   WHEN {kifejezés | feltétel}
     THEN utasítás [utasítás]…
   [WHEN {kifejezés | feltétel}
     THEN utasítás [utasítás]…]…
   [ELSE utasítás [utasítás]…]
END CASE;
```

CASE elágaztató utasítás

- A CASE egy olyan elágaztató utasítás, ahol az egymást kölcsönösen kizáró tevékenységek közül egy kifejezés értékei, vagy feltételek teljesülése szerint lehet választani.
- Tehát egy CASE utasítás tetszőleges számú WHEN ágból és egy opcionális ELSE ágból áll.
- Ha a szelektor_kifejezés szerepel, akkor a WHEN ágakban kifejezés áll, ha nem szerepel, akkor feltétel.

CASE elágaztató utasítás működése

- Ha szerepel szelektor_kifejezés,
  - akkor ez kiértékelődik,
  - majd az értéke a felírás sorrendjében hasonlításra kerül a WHEN ágak kifejezéseinek értékeivel.
  - Ha megegyezik valamelyikkel, akkor az adott ágban a THEN után megadott utasítássorozat hajtódik végre.
  - Ha a szelektor_kifejezés értéke nem egyezik meg egyetlen kifejezés értékével sem és van ELSE ág, akkor végrehajtódnak az abban megadott utasítások.
  - Ha viszont nincs ELSE ág, akkor a CASE_NOT_FOUND kivétel váltódik ki.

CASE elágaztató utasítás működése

- Ha a CASE alapszó után nincs megadva szelektor_kifejezés,
  - akkor a felírás sorrendjében sorra kiértékelődnek a feltételek és amelyik igaz értéket vesz fel, annak a WHEN ága kerül kiválasztásra.
  - A szemantika a továbbiakban azonos a fent leírtakkal.

CASE elágaztató utasítás

- Példa:

```
BEGIN
  CASE 2
    WHEN 1 THEN
      DBMS_OUTPUT.PUT_LINE('2 = 1');
    WHEN 1+2 THEN
      DBMS_OUTPUT.PUT_LINE('2 = 1 + 2 ');
    ELSE
      DBMS_OUTPUT.PUT_LINE('egyik sem');
  END CASE;
END;
```

CASE elágaztató utasítás

- Példa:

```
DECLARE
  v_szam NUMBER := 10;
BEGIN
  CASE
    WHEN v_szam MOD 2 = 0
      THEN DBMS_OUTPUT.PUT_LINE('Páros.');
    WHEN v_szam < 5
      THEN DBMS_OUTPUT.PUT_LINE('Kisebb 5-nél.');
    WHEN v_szam > 5
      THEN DBMS_OUTPUT.PUT_LINE('Nagyobb 5-nél.');
    ELSE DBMS_OUTPUT.PUT_LINE('Ez csak az 5 lehet.');
  END CASE;
END;
```

Ciklusok

- Alapciklus (végtelen ciklus)
- WHILE ciklus
- FOR ciklus
- Kurzor FOR ciklus (később részletezve)

- Ciklusból kilépés: `EXIT [WHEN feltétel];`
- Ciklus folytatása: `CONTINUE [WHEN feltétel];`

Ciklusok

- A ciklusmag ismétlődésére vonatkozó információkat (amennyiben vannak) a mag előtt, a ciklus fejében kell megadni.
- Ezek az információk az adott ciklusfajtára nézve egyediek.
- Egy ciklus a működését mindig csak a mag első utasításának végrehajtásával kezdheti meg. Egy ciklus befejeződhet, ha
  - az ismétlődésre vonatkozó információk ezt kényszerítik ki;
  - GOTO utasítással kilépünk a magból;
  - az EXIT utasítással befejeztetjük a ciklust;
  - kivétel váltódik ki.

Alapciklus (végtelen ciklus)

- Alakja:

```
[címke] LOOP
          utasítás [utasítás]…
        END LOOP [címke];
```

- Példa:

```
DECLARE
  a NUMBER(5) := 10; b NUMBER(5) := 10;
BEGIN
  LOOP
    b := b / a;
    a := a - 1;
  END LOOP;
END;
```

Alapciklus (végtelen ciklus)

- Az alapciklusnál nem adunk információt az ismétlődésre vonatkozóan, tehát ha a magban nem fejeztetjük be a másik három lehetőség valamelyikével, akkor végtelenszer ismétel.

WHILE ciklus

- Alakja:

```
[címke] WHILE feltétel
        LOOP utasítás [utasítás]…
        END LOOP [címke];
```

- A WHILE ciklus működésének két szélsőséges esete van:
  - Ha a feltétel a legelső esetben hamis vagy NULL, akkor a ciklusmag egyszer sem fut le (üres ciklus).
  - Ha a feltétel a legelső esetben igaz és a magban nem történik valami olyan, amely ezt az értéket megváltoztatná, akkor az ismétlődés nem fejeződik be (végtelen ciklus).

WHILE ciklus

- Példa:

```
DECLARE
    v_faktorialis NUMBER(5); i PLS_INTEGER;
BEGIN
  i := 1;
  v_faktorialis := 1;
  WHILE v_faktorialis * i < 10**5 LOOP
    v_faktorialis := v_faktorialis * i;
    i := i + 1;
  END LOOP;
END;
```

FOR ciklus

- Alakja:

```
[címke]
FOR ciklusváltozó IN [REVERSE] alsó_határ..felső_határ
LOOP
  utasítás [utasítás]…
END LOOP [címke];
```

FOR ciklus

- A ciklusváltozó egy implicit módon PLS_INTEGER típusúnak deklarált változó, amelynek hatásköre a ciklusmag.
  - Ez a változó rendre felveszi az alsó_határ és felső_határ által meghatározott egész tartomány minden értékét és minden egyes értékére egyszer lefut a mag.
  - A ciklusváltozónak a ciklus magjában nem lehet értéket adni, csak az aktuális értékét lehet felhasználni kifejezésben.
- Az alsó_határ és felső_határ egész értékű kifejezés lehet. A kifejezések egyszer, a ciklus működésének megkezdése előtt értékelődnek ki.
- A REVERSE kulcsszó megadása esetén a ciklusváltozó a tartomány értékeit csökkenően, annak hiányában növekvően veszi fel.
  - A REVERSE megadása esetén is a tartomány alsó határát kell először megadni.

FOR ciklus

- Példa:

```
DECLARE
  v_osszeg NUMBER(10):=0;
BEGIN
  FOR i IN REVERSE 1..100
  LOOP
    v_osszeg := v_osszeg + i;
  END LOOP;
END;
```

EXIT utasítás

- Alakja:

```
EXIT [címke] [WHEN feltétel];
```

- Az EXIT utasítás bármely ciklus magjában kiadható, de ciklusmagon kívül nem használható.
- Az EXIT hatására a ciklus működése befejeződik és a program a következő utasításon folytatódik.
- Az EXIT címke utasítással az egymásba skatulyázott ciklusok sorozatát fejeztetjük be a megadott címkéjű ciklussal bezárólag.
- A WHEN utasításrész a ciklus feltételes befejeződését eredményezi.
  - Csak a feltétel igaz értéke mellett nem folytatódik tovább az ismétlődés.

EXIT utasítás

- Példa:

```
DECLARE
  v_osszeg PLS_INTEGER := 0;
BEGIN
<<kulso>>
  FOR i IN 1..100 LOOP
    FOR j IN 1..i LOOP
      v_osszeg := v_osszeg + i;
      EXIT kulso WHEN v_osszeg > 100;
    END LOOP;
  END LOOP;
END;
```

CONTINUE utasítás

- Alakja:

```
CONTINUE [címke] [WHEN feltétel];
```

- A CONTINUE utasítás bármely ciklus magjában kiadható, de ciklusmagon kívül nem használható.
- A CONTINUE hatására kimaradnak a ciklusmag hátralévő utasításai, és a ciklus a következő iterációs lépéssel folytatódik.
- A CONTINUE címke utasítással az egymásba skatulyázott ciklusok közül az adott címkéjű ciklus folytatódik a következő iterációs lépéssel.
- Ha a WHEN utasításrész meg van adva, akkor a CONTINUE utasítás csak akkor lép a következő iterációs lépésre, ha a feltétel igaz.

CONTINUE utasítás

- Példa:

```
DECLARE
  x NUMBER := 0;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('x=' || x);
    x := x + 1;
    CONTINUE WHEN x < 3;
    DBMS_OUTPUT.PUT_LINE('After CONTINUE: x=' || x);
    EXIT WHEN x = 5;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('After loop: x=' || x);
END;
```