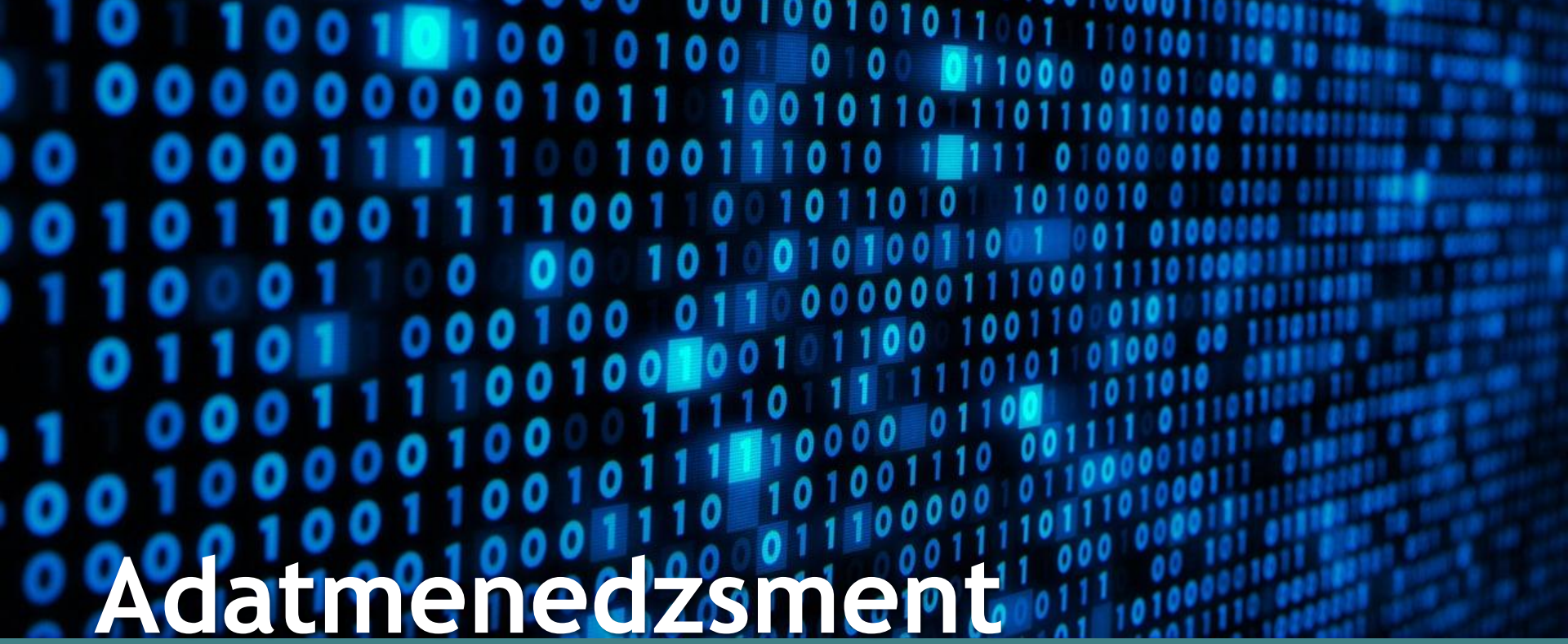

Adatmenedzsment

7. gyakorlat

Kurzor FOR ciklus

- A kurzor FOR ciklus általános alakja:

```
FOR ciklusváltozó IN (select_utasítás)
LOOP
    utasítás [, utasítás]…
END LOOP;
```

- Az explicit kurzor használatát a kurzor FOR ciklus alkalmazásával egyszerűsíthetjük.
- Implicit módon `kurzornév%ROWTYPE` típusú lesz a ciklusváltozó:
  - Megnyitja a kurzort
  - Betölti az aktív halmaz összes sorát
  - Lezárja a kurzort

Kurzor FOR ciklus - példa

```
BEGIN
  FOR c IN (SELECT * FROM LOCATIONS
            WHERE COUNTRY_ID='US')
  LOOP
    DBMS_OUTPUT.PUT_LINE(c.location_id ||
                         ' ' || c.city);
  END LOOP;

END;
```

PL/SQL alprogramok

- Egy PL/SQL alprogram egy nevesített PL/SQL blokk, amelyet paraméterek halmazával lehet meghívni, ha vannak paraméterei.
- Lehet eljárás vagy függvény:
  - Az eljárásokat általában egy művelet végrehajtására használjuk.
  - A függvényt általában egy érték kiszámítására és visszaadására használjuk.
- A PL/SQL alprogramok szerepelhetnek:
  - Blokkba ágyazva (ami lehet egy másik alprogram is)
  - Séma szinten
  - Csomagban

PL/SQL alprogramok

- Az alprogramok deklarációját csak az összes többi objektum deklarációja után helyezhetjük el.
  - Pragma még következhet utána.
- Az alprogramok rekurzívan hívhatóak.
- Az alprogramoknak két fő része van: a specifikáció és a törzs.
  - Az alprogram törzse a DECLARE alapszó elhagyásától eltekintve megegyezik egy névtelen blokkal.

PL/SQL alprogramok

- PL/SQL eljárások általános alakja:

```
PROCEDURE név[(formális paraméterlista)] IS
[deklarációs utasítás(ok)]
BEGIN
    végrehajtható utasítás(ok)
    [EXCEPTION
        kivételkezelő]
END [név];
```

PL/SQL alprogramok

- Eljárások:
  - Az eljárást utasításszerűen tudjuk meghívni.
  - Eljáráshívás szerepelhet a programban bárhol, ahol végrehajtható utasítás állhat.
  - A hívás formája: név és aktuális paraméter lista.
  - Az eljárás befejeződik, ha elfogynak a végrehajtható utasításai, vagy pedig a RETURN utasítás hajtódik végre. Ekkor a vezérlés visszatér a hívást követő utasításra.
  - Eljárás nem fejeztethető be GOTO utasítással.

PL/SQL alprogramok

- PL/SQL függvények általános alakja:

```
FUNCTION név[(formális paraméterlista)]
RETURN típus IS
[deklarációs utasítás(ok)]
BEGIN
    végrehajtható utasítás(ok)
    [EXCEPTION
        kivételkezelő]
END [név];
```

PL/SQL alprogramok

- Függvények:
  - A specifikációban a RETURN alapszó után a típus a függvény visszatérési értékének típusát határozza meg.
  - Egy függvényt meghívni kifejezésben lehet.
  - A hívás formája: név és aktuális paraméter lista.
  - A függvény törzsében legalább egy RETURN utasításnak szerepelnie kell, különben a PROGRAM_ERROR kivétel váltódik ki a működése közben.
  - A függvényben használt RETURN utasítás a visszatérési értéket is meghatározza. Alakja: `RETURN [ ( ] kifejezés [ ) ];`

Formális paraméterlista

- Formális paraméterekből áll:

```
név [ {IN | {OUT | IN OUT} [NOCOPY]} ] típus
[ { := | DEFAULT } kifejezés ]
```

- IN: érték szerinti paraméterátadás (de nincs értékmásolás).
- OUT: eredmény szerinti paraméterátadás.
- IN OUT: érték-eredmény szerinti paraméterátadás.
- NOCOPY: ajánlás (hint) a fordítónak, hogy OUT és IN OUT esetben se másoljon értéket.
- A paraméter-összerendelés történhet pozíció és/vagy név szerint.
- A lokális és csomagbeli alprogramnevek túlterhelhetők.

Formális paraméterlista

- Az IN, OUT és az IN OUT a paraméterátadás módja. Ha nem adjuk meg, akkor alapértelmezett az IN.
- Az alprogram törzsében az IN módú paraméter nevesített konstansként, az OUT módú változóként, az IN OUT módú pedig inicializált változóként kezelhető.
  - Tehát az IN módú paraméternek nem adható érték!
- Az OUT módú formális paraméter automatikus kezdőértéke NULL.
- IN mód esetén az aktuális paraméter kifejezés, OUT és IN OUT esetén változó lehet.
- Az IN módú formális paramétereknek kezdőérték adható egy kifejezés segítségével (:= vagy DEFAULT után).

Példa – IN mód

```
DECLARE
    v_num NUMBER;
    PROCEDURE in_num(p_in IN NUMBER) IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE(p_in);
    END in_num;
BEGIN
    v_num := 10;
    FOR I IN 1..10 LOOP
        in_num(v_num * i);
    END LOOP;
END;
```

Példa – OUT mód

```
DECLARE
    v_num NUMBER;
    PROCEDURE out_num(p_out OUT NUMBER) IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE(NVL(p_out, -1));
        p_out := 20;
    END out_num;
BEGIN
    v_num := 10;
    out_num(v_num);
    DBMS_OUTPUT.PUT_LINE(v_num);
END;
```

Példa – IN OUT mód

```
DECLARE
    v_num NUMBER;
    PROCEDURE inout_num(p_inout IN OUT NUMBER) IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE(p_inout);
        p_inout := 30;
    END inout_num;
BEGIN
    v_num := 10;
    inout_num(v_num);
    DBMS_OUTPUT.PUT_LINE(v_num);
END;
```

Formális és aktuális paraméterek

- Egymáshoz rendelés:
  - Pozíció szerinti (sorrendi kötés): a formális és aktuális paraméterek sorrendje a döntő.
  - Név szerinti kötés: az aktuális paraméter listán a formális paraméterek sorrendjétől függetlenül, tetszőleges sorrendben felsoroljuk a formális paraméterek nevét, majd a => jelkombináció után megadjuk a megfelelő aktuális paramétert.
  - Keverve a kettőt: először a pozíció szerinti majd a név szerinti egymáshoz rendeléseket megadva.
- Az alprogramok paraméterszáma rögzített.

Formális és aktuális paraméterek

- Az aktuális paraméterek száma lehet kevesebb, mint a formális paraméterek száma:
  - Ez az IN módú paraméterek kezdőértékétől függ.
  - Egy olyan formális paraméterhez, amelynek van kezdőértéke, nem szükséges megadnunk aktuális paramétert.
  - Ha megadunk, akkor az inicializálás az aktuális paraméter értékével, ha nem adunk meg, akkor a formális paraméter listán szereplő kezdőértékkel történik.

Formális és aktuális paraméterek

- Alprogramok túlterhelése:
  - Az ilyen alprogramoknál a név azonos, de a formális paraméterek száma, típusa, vagy sorrendje eltérő kell legyen.
  - Ekkor a fordító az aktuális paraméter lista alapján választja ki a hívásnál a megfelelő törzset.

Sémaszintű alprogramok

- Az alprogramokat kezelhetjük adatbázis-objektumokként:
  - Ekkor az SQL szokásos létrehozó, módosító és törlő DDL utasításait használhatjuk.
  - A tárolt alprogramok lefordított formában, ún. M-kódban tárolódnak.

Sémaszintű alprogramok

- Tárolt eljárás létrehozása:

```
CREATE [OR REPLACE] eljárásfej
[AUTHID {DEFINER|CURRENT_USER}]
eljárás_törzs
```

- Tárolt függvény létrehozása:

```
CREATE [OR REPLACE] függvényfej
[AUTHID {DEFINER|CURRENT_USER}] [DETERMINISTIC]
függvénytörzs
```

Sémaszintű alprogramok

- Az OR REPLACE rész segítségével újragenerálódik az eljárás vagy függvény, ha már létezik.
  - Az alprogram definíciójának megváltoztatására szolgál, így az alprogram előzőleg már megadott objektum-jogosultságokat nem kell törölni, újra létrehozni és újra adományozni.
- Az AUTHID segítségével megadható, hogy az alprogram létrehozójának (DEFINER – ez az alapértelmezés), vagy aktuális hívójának (CURRENT_USER) jogosultságai érvényesek-e a hívásnál.

Sémaszintű alprogramok

- A DETERMINISTIC egy optimalizálási előírás, amely a redundáns függvényhívások elkerülését szolgálja. Megadása esetén a függvény visszatérési értékéről egy másolat készül, és ha a függvény ugyanazokkal az aktuális paraméterekkel kerül meghívásra, az optimalizáló ezt a másolatot fogja használni.

Sémaszintű alprogramok

- Tárolt eljárás újrafordítása:

```
ALTER PROCEDURE eljárásnév
COMPILE [DEBUG];
```

- Tárolt függvény újrafordítása:

```
ALTER FUNCTION függvénynév
COMPILE [DEBUG];
```

- Tárolt eljárás eldobása:

```
DROP PROCEDURE eljárásnév;
```

- Tárolt függvény eldobása:

```
DROP FUNCTION függvénynév;
```