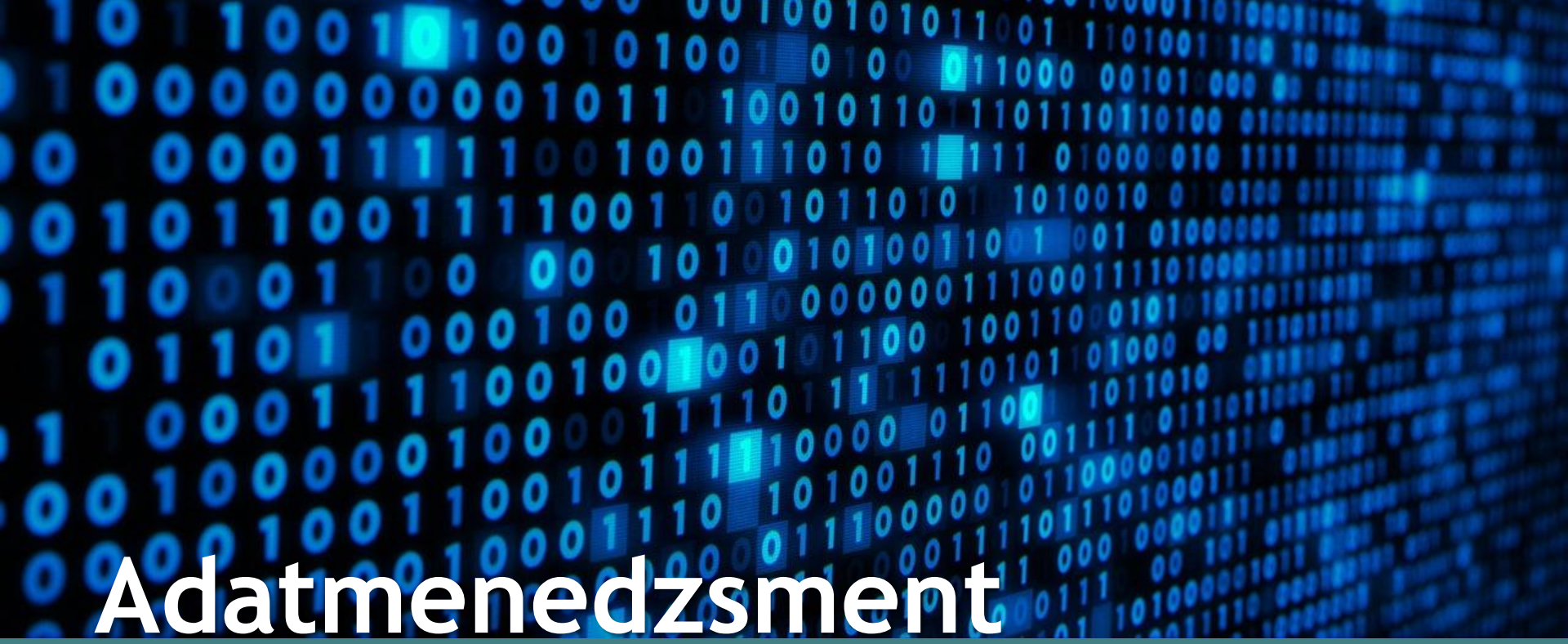

Adatmenedzsment

6. gyakorlat

FORALL

- Általános alakja:

```
FORALL index IN {alsó_határ..felső_határ | INDICES OF
kollekció [BETWEEN alsó_határ AND felső_határ] | VALUES
OF indexkollekció_név}
[SAVE EXCEPTIONS]
sql_utasítás;
```

- Az SQL motor az SQL utasítást a megadott indextartomány minden értéke mellett egyszer végrehajtja. (Az adott indexű kollekcióelemeknek léteznie kell.)
- Az `sql_utasítás` egy olyan INSERT, DELETE vagy UPDATE utasítás, amely kollekcióelemeket hivatkozik a WHERE, VALUES, vagy SET utasításrészében.

FORALL

- A FORALL a PL/SQL-oldali együttes hozzárendelés eszköze.
- A FORALL implicit módon a kollekció indexének megfelelő típusúnak deklarálja az indexet.
  - `INDICES OF kollekció`: a megadott kollekció elemeinek az indexeit veszi fel.
    - `BETWEEN`-nel korlátozhatjuk.
  - `VALUES OF indexkollekció_név`: az index által felveendő értékeket az `indexkollekció_név` által megnevezett kollekció tartalmazza.

FORALL - példa

```
CREATE TABLE EMPLOYEES_TMP AS SELECT * FROM EMPLOYEES;

DECLARE
    TYPE t_num_list IS VARRAY(20) OF NUMBER;
    depts t_num_list := t_num_list(10, 30, 70);

BEGIN
    FORALL i IN depts.FIRST .. depts.LAST
        DELETE FROM EMPLOYEES_TMP
        WHERE DEPARTMENT_ID = depts(i);
END;
```

FORALL

- Ha egy FORALL utasításban az SQL utasítás egy nem kezelt kivételt vált ki, akkor az egész FORALL visszagörgetődik.
- Ha viszont a kiváltott kivételt kezeljük, akkor csak a kivételt okozó végrehajtás görgetődik vissza az SQL utasítás előtt elhelyezett implicit mentési pontig, a korábbi végrehajtások eredménye megmarad.

FORALL - példa

```
CREATE TABLE FORALL_EXC (NUM NUMBER(1));

DECLARE
    TYPE t_nt IS TABLE OF NUMBER(5);
    v_nt t_nt := t_nt(1, 5, 30);

BEGIN
    FORALL I IN INDICES OF V_NT
        INSERT INTO FORALL_EXC VALUES (v_nt(I));
    EXCEPTION WHEN OTHERS THEN NULL;

END;
```

Kurzorok

- Környezeti terület: egy memóriaterület, amelynek tartalma:
  - információ a feldolgozott sorokról
    - lekérdezés esetén tartalmazza a visszaadott sorokat, amelyek az aktív halmazt alkotják
  - mutató az utasítás belső reprezentációjára
- A kurzor egy olyan eszköz,
  - mellyel megnevezhetjük a környezeti területet,
  - hozzáférhetünk az ott elhelyezett információkhoz,
  - és amennyiben az aktív halmaz több sort tartalmaz, azokat egyenként elérhetjük, feldolgozhatjuk.

Kurzorok

- Kurzorok fajtái:
  - explicit: A több sort (pontosabban akármennyi sort) visszaadó lekérdezések eredményének kezelésére szolgál.
  - implicit: a PL/SQL automatikusan felépít egy implicit kurzort minden DQL és DML utasításhoz.
  - rejtett: kurzor FOR ciklus esetén (*lásd később*).

Explicit kurzor - életciklus

- 1. deklaráció

```
CURSOR név[(paraméter[, paraméter]…)]
[RETURN sortípus] IS select_utasítás;
```

- 2. megnyitás

```
OPEN kurzornév[(aktuális_paraméterlista)];
```

- 3. sorok betöltése

```
FETCH kurzornév
INTO {rekordnév | változónév[, változónév]…} |
BULK COLLECT INTO kollekciónév [, kollekciónév]…
[LIMIT sorok];
```

- 4. lezárás

```
CLOSE kurzornév;
```

Explicit kurzor - életciklus

- Megnyitás:
  - A kurzor megnyitásánál lefut a kurzorhoz rendelt lekérdezés, meghatározódik az aktív halmaz és az aktív halmazhoz rendelt kurzormutató az első sorra áll rá.
- Megnyitott kurzort nem lehet újra megnyitni. Megnyitott kurzorra kiadott OPEN utasítás a CURSOR_ALREADY_OPEN kivételt váltja ki.
- Megnyitott kurzor neve nem szerepeltethető kurzor FOR ciklusban.

Explicit kurzor - életciklus

- Betöltés:
  - Az aktív halmaz sorainak feldolgozását a FETCH utasítás teszi lehetővé.
  - A FETCH utasítás az adott kurzorhoz tartozó kurzormutató által címzett sort betölti a rekordba, vagy a megadott skalárváltozókba, és a kurzormutatót a következő sorra állítja.
  - A skalárváltozókba a sor oszlopainak értéke kerül, a változók és oszlopok típusának kompatibilisnek kell lenniük. Rekord megadása esetén az oszlopok és mezők típusának kell kompatibilisnek lennie.
  - A skalárváltozók száma, illetve a rekord mezőinek száma meg kell egyezzen az oszlopok számával.

Explicit kurzor - életciklus

- Betöltés:
  - Nem megnyitott kurzor vagy kurzorváltozó esetén a FETCH utasítás az INVALID_CURSOR kivételt váltja ki.
  - Ha a FETCH utasítást az utolsó sor feldolgozása után adjuk ki, akkor a változók vagy rekord előző értéke megmarad.
  - Nem létező sor betöltése nem vált ki kivételt.
    - Ezen szituáció ellenőrzésére használjuk a %FOUND és a %NOTFOUND attribútumokat.

Explicit kurzor - életciklus

- Lezárás:
  - A kurzor lezárása érvényteleníti a kurzor vagy kurzorváltozó és az aktív halmaz közötti kapcsolatot és megszünteti a kurzormutatót.
  - Lezárni csak megnyitott kurzort lehet, különben az INVALID_CURSOR kivétel váltódik ki.

Explicit kurzor - példák

```
CURSOR cur_emps RETURN EMPLOYEES%ROWTYPE IS
SELECT * FROM EMPLOYEES ORDER BY
UPPER(LAST_NAME);

v_id LOCATIONS.LOCATION_ID%TYPE;
CURSOR cur_locs IS
SELECT CITY, POSTAL_CODE FROM LOCATIONS
WHERE LOCATION_ID = v_id;

CURSOR cur_jobs(p_min_sal JOBS.MIN_SALARY%TYPE
:= 8000) IS
SELECT * FROM JOBS
WHERE MIN_SALARY >= p_min_sal;
```

Kurzor attribútumok

- A kurzor attribútumok a DQL és DML utasítás végrehajtásáról szolgáltatnak információkat. Csak procedurális utasításokban használhatók, SQL utasításokban nem.
- **%FOUND**: megnyitás után, de első betöltés előtt NULL, sikeres betöltés esetén TRUE, egyébként FALSE.
- **%ISOPEN**: ha meg van nyitva a kurzor TRUE, egyébként FALSE.
- **%NOTFOUND**: a %FOUND negáltja.
- **%ROWCOUNT**: megnyitás után, de első betöltés előtt 0, minden sikeres betöltés esetén eggyel nő.
- Ha az explicit kurzor nincs megnyitva, a %FOUND, %NOTFOUND és %ROWCOUNT alkalmazása az INVALID_CURSOR kivételt váltja ki.

Explicit kurzor - példák

```
DECLARE
    CURSOR cur IS SELECT * FROM employees WHERE job_id
    LIKE '%MGR' OR job_id LIKE '%MAN' ORDER BY job_id;
    v_employees employees%ROWTYPE;

BEGIN
    OPEN cur;
    LOOP
        FETCH cur INTO v_employees;
        EXIT WHEN cur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_employees.last_name
                    || ' ' || v_employees.job_id);
    END LOOP;
    CLOSE cur;

END;
```

Explicit kurzor - példák

```
DECLARE
    CURSOR c1 IS SELECT LAST_NAME FROM EMPLOYEES;
    name EMPLOYEES.LAST_NAME%TYPE;

BEGIN
    OPEN c1;
    LOOP
        FETCH c1 INTO name;
        EXIT WHEN c1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(c1%ROWCOUNT || '. ' || name);
        IF c1%ROWCOUNT mod 5 = 0 THEN
            DBMS_OUTPUT.PUT_LINE('--- Fetched 5 rows ---');
        END IF;
    END LOOP;
    CLOSE c1;

END;
```

Kurzorok - kivételek

- CURSOR_ALREADY_OPEN: megnyitott kurzor újra megnyitásakor
- INVALID_CURSOR: nem megnyitott kurzoron végzett FETCH, CLOSE vagy kurzor attribútum hivatkozás (kivéve az %ISOPEN-t) esetén