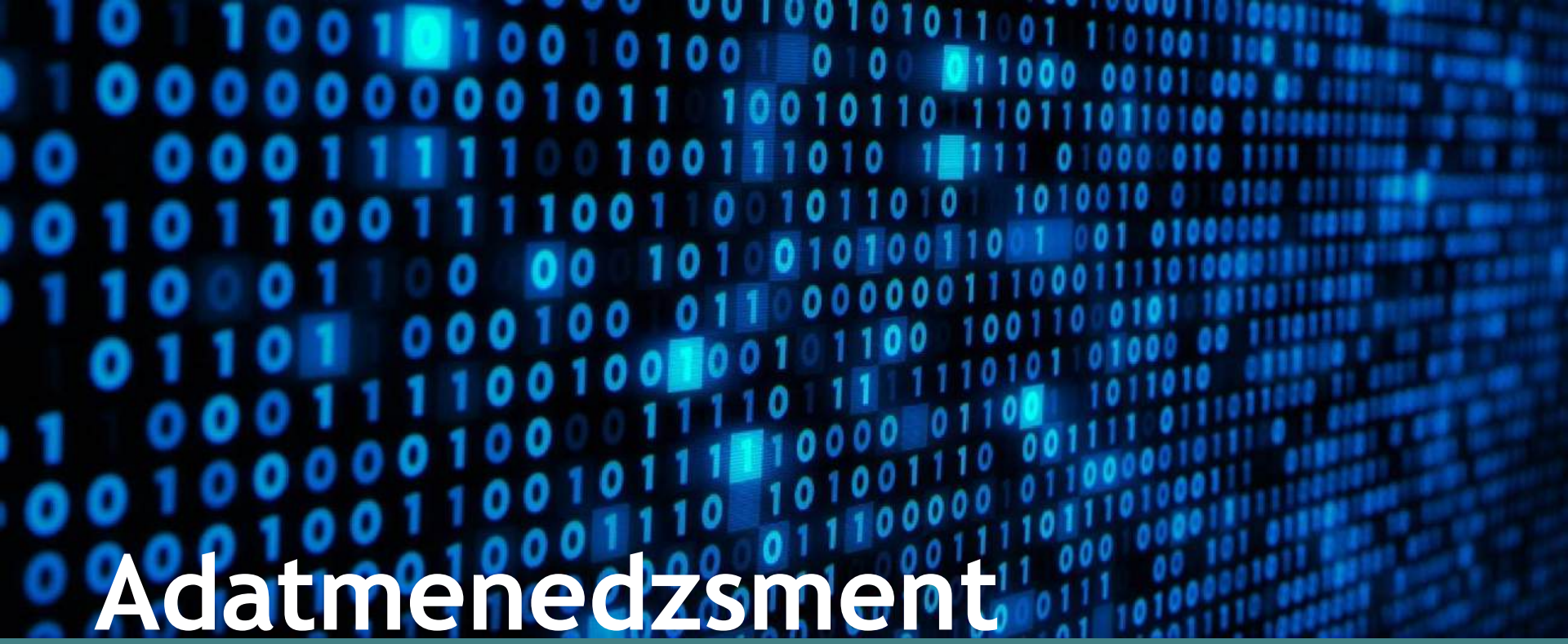

Adatmenedzsment

3. gyakorlat

Összetett adattípusok

- Összetett adattípusok PL/SQL-ben:
  - Rekordtípus
  - Kollekciótípus (*lásd később*)
    - asszociatív tömb
    - beágyazott tábla
    - dinamikus tömb

Rekordtípus

- A rekord logikailag egybetartozó adatok egy heterogén csoportja, ahol minden adatot egy-egy mező tárol.
- A mezőnek saját *neve* és *típusa* van.
- A rekordtípus lehetővé teszi különböző adatok együttesét egyetlen logikai egységként kezeljünk.
- A rekord adattípus segítségével olyan programeszközöket tudunk deklarálni, amelyek egy adatbázistábla sorait tudják közvetlenül kezelni.

Rekordtípus

- Rekordtípus definíció:

```
TYPE név IS RECORD (
    mezőnév típus [[NOT NULL] {:=|DEFAULT} kifejezés]
    [, mezőnév típus [[NOT NULL] {:=|DEFAULT} kifejezés]]…
);
```

- Egy rekord deklarációja:
  - `rekordnév rekordtípusnév;`
- Hivatkozás a rekord mezőjére:

```
rekordnév.mezőnév;
```

Rekordtípus

- Példa:

```
TYPE t_alk_rec IS RECORD (
    azon    NUMBER(5) NOT NULL,
    nev     VARCHAR2(50) NOT NULL,
    fizetes NUMBER(8,2),
    email   VARCHAR2(30)
);
```

- Deklaráció:

```
v_fonok t_alk_rec;
```

- Hivatkozás:

```
v_fonok.nev
```

%TYPE

- A **%TYPE** egy korábban már deklarált kollekció, kurzorváltozó, mező, objektum, rekord, adatbázistábla oszlop vagy változó típusát veszi át és ezzel a típussal deklarálja a változót vagy nevesített konstanst.
- A **%TYPE** használatának két előnye van:
  - nem kell pontosan ismerni az átvett típust,
  - ha az adott eszköz típusa megváltozik, a változó típusa is automatikusan, futási időben követi ezt a változást.

%TYPE

- A **%TYPE** használatával öröklődik:
  - a típus,
  - a méret és
  - a megszorítások (ha a hivatkozott elem **nem oszlop**)
- A **%TYPE** használatával nem öröklődik a kezdőérték.

%ROWTYPE

- A **%ROWTYPE** lehetővé teszi olyan rekordváltozó deklarálását, amely egy adatbázistábla vagy nézet teljes vagy részleges sorát reprezentálja.
  - Azaz minden sor minden látható oszlopához a rekordnak van egy ugyanolyan nevű és típusú mezője.
- Ha a sor struktúrája változik, akkor a rekord struktúrája is ennek megfelelően változik.
- A rekord mezői **nem öröklik** az oszlopok megszorításait és kezdőértékeit.

SQL utasítások PL/SQL-ben

- Egy PL/SQL program szövegébe közvetlenül beépíthetők az SQL DQL, DML és TCL utasításai.
- A DQL, DML és TCL utasítások bárhol használhatók, ahol végrehajtható utasítások állhatnak.
- A statikus SQL utasítások szövege a fordításkor ismert, a PL/SQL-fordító ugyanúgy kezeli, fordítja őket, mint a procedurális utasításokat.
  - A PL/SQL egy külön eszközt használ arra, hogy minden SQL-utasítást alkalmazhassunk egy PL/SQL-programban, ez a natív dinamikus SQL (*lásd később*).

SELECT INTO

- Általános alakja:

```
SELECT …
INTO {változónév[, változónév]… | rekordnév}
FROM táblahivatkozás
további utasításrészek;
```

- Példa:

```
DECLARE v_salary EMPLOYEES.SALARY%TYPE;
BEGIN
  SELECT SALARY INTO v_salary FROM EMPLOYEES
  WHERE EMPLOYEE_ID = 100;
END;
```

SELECT INTO

- A SELECT INTO utasítás egy vagy több adatbázistáblát kérdez le és a származtatott értékeket változókba vagy egy rekordba helyezi el.
- Az INTO utasításrészben minden kifejezéshez meg kell adni egy típus kompatibilis változót, vagy pedig a rekordnak rendelkeznie kell megfelelő mezőkkel.
- Egy ilyen SELECT utasításnak pontosan egy sort kell visszaadnia.
- Lehetséges kivételek:
  - TOO_MANY_ROWS – ha az eredményhalmaz egynél több sorból áll
  - NO_DATA_FOUND – ha az eredményhalmaz üres

DELETE

- Általános alakja:

```
DELETE [FROM] táblahivatkozás
[WHERE feltétel]
[returning_utasításrész];
```

- A returning utasításrész:

```
RETURNING
egysoros_select_kif[, egysoros_select_kif]…
INTO {változó[, változó]… | rekord}
```

DELETE

- A returning utasításrész segítségével a törölt sorok alapján számított értékek kaphatók vissza.
  - Ebben az esetben nem szükséges ezeket az értékeket a törlés előtt egy SELECT segítségével származtatni.
- Az értékek változókban, rekordban tárolhatók le.
- Példa:

```
DECLARE v_salary EMPLOYEES.SALARY%TYPE;
BEGIN
    DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID = 206
    RETURNING SALARY INTO v_salary;
END;
```

INSERT

- Általános alakja:

```
INSERT INTO táblahivatkozás
  [(oszlop[, oszlop]…)]
VALUES
  {(kifejezés[, kifejezés]…) | rekord}
[returning_utasításrész];
```

- A returning utasításrész:

```
RETURNING
egysoros_select_kif[, egysoros_select_kif]…
INTO {változó[, változó]… | rekord}
```

UPDATE

- Általános alakja:

```
UPDATE táblahivatkozás
SET oszlop = kifejezés
[, oszlop = kifejezés]…
[WHERE feltétel]
[returning_utasításrész];
```

- A returning utasításrész:

```
RETURNING
egysoros_select_kif[, egysoros_select_kif]…
INTO {változó[, változó]… | rekord}
```