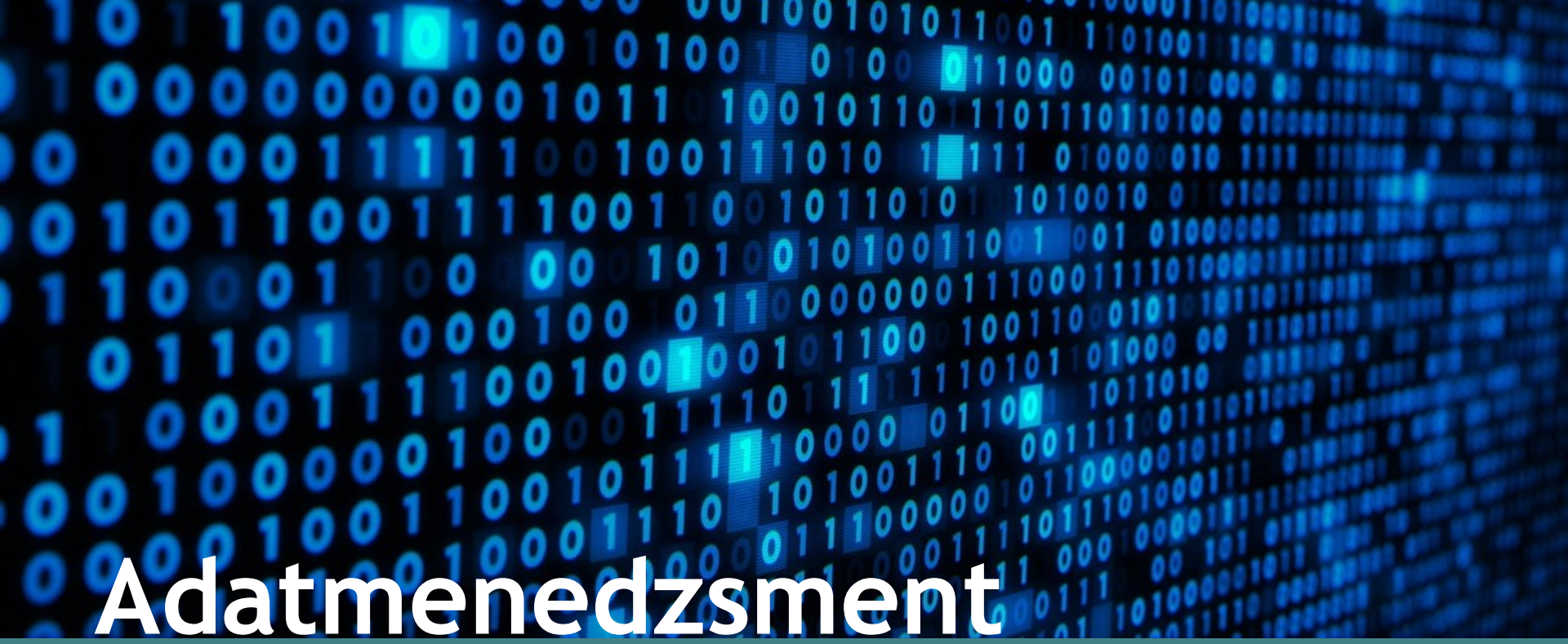

Adatmenedzsment

4. gyakorlat

Kivételkezelés

- Futási időben bekövetkező kivételek fajtái:
  - beépített (azaz a futtató rendszer váltja ki):
    - előre definiált
    - nem előre definiált
  - felhasználói (explicit módon kerülnek kiváltásra)
- Az előre definiált és a felhasználói kivételeknek van nevük.
- Minden beépített kivételnek van kódja és szövege.
- Két beépített függvény:
  - SQLCODE: a legutoljára bekövetkezett kivétel kódja
  - SQLERRM[(*k*)]: a legutoljára bekövetkezett (ill. a *k* kódú) kivételhez tartozó szöveg (hibaüzenet)

SQLCODE

- Az SQLCODE lehetséges értékei:
  - 0, ha nem történt kivétel,
  - 1, ha felhasználói kivétel történt,
  - +100, ha a NO_DATA_FOUND beépített kivétel következett be,
  - egy negatív szám, bármely más beépített kivétel esetén.

A PL/SQL blokk

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

Kivételkezelő

- A kivétel kezelő rész általános alakja:

```
EXCEPTION
    WHEN kivételnév [OR kivételnév]…
        THEN utasítás [utasítás]…
    [WHEN kivételnév [OR kivételnév]…
        THEN utasítás [utasítás]…]
    [WHEN OTHERS
        THEN utasítás [utasítás]…]
```

A kivételkezelő működése

- Ha bekövetkezik egy kivétel a végrehajtható részben:
  - a futás félbeszakad,
  - a futtató rendszer megnézi, hogy van-e a blokk végén kivételkezelő, és annak valamely WHEN ága nevesíti-e a bekövetkezett kivételt:
    - ha van, akkor lefutnak a THEN utáni utasítások
    - különben, ha van WHEN OTHERS ág, akkor az abban lévő utasítások futnak le
  - ha nincs kivételkezelő, akkor a kivétel továbbadódik az aktiváló környezetnek
  - ha nincs ilyen, akkor az UNHANDLED_EXCEPTION váltódik ki
- Ha a kivétel a deklarációs részben vagy a kivételkezelőben váltódik ki kivétel, akkor az azonnal továbbadódik a környezetnek.

A kivételkezelő működése

- Előre definiált kivételek esetében hivatkozunk a kivétel nevére:
  - `EXCEPTION WHEN VALUE_ERROR THEN …`
- Nem előre definiált kivételek esetében a deklarációs részben:
  - Deklarálunk egy kivételnevet: `név EXCEPTION;`
  - Összerendeljük egy kóddal: `PRAGMA EXCEPTION_INIT(név, kód);`
  - Majd a kivételkezelőben hivatkozunk a nevére
- A felhasználó által definiált kivételek esetében:
  - A deklarációs részben deklarálunk egy kivételnevet
  - A végrehajtható részben kiváltjuk a kivételt (RAISE) ahol szükséges
  - A kivételkezelőben hivatkozunk a nevére

A kivételkezelő működése

- Példa – nem előre definiált kivételek:

(Megjegyzés: ORA-01400: cannot insert NULL)

```
DECLARE insert_null_exc EXCEPTION;
PRAGMA EXCEPTION_INIT(insert_null_exc, -1400);
BEGIN
  INSERT INTO DEPARTMENTS(DEPARTMENT_ID, DEPARTMENT_NAME)
  VALUES (999, NULL);
EXCEPTION
  WHEN insert_null_exc THEN
    DBMS_OUTPUT.PUT_LINE('Cannot insert NULL value.');
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

A kivételkezelő működése

- Példa - felhasználó által definiált kivételek:

```
DECLARE too_few_emps EXCEPTION; v_count PLS_INTEGER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM EMPLOYEES WHERE
  DEPARTMENT_ID = 50;
  IF v_count < 10 THEN
    RAISE too_few_emps;
  END IF;
  EXCEPTION
    WHEN too_few_emps THEN
      DBMS_OUTPUT.PUT_LINE('There are too few employees.');
END;
```

RAISE

- Bármely megnevezett kivétel kiváltható a `RAISE kivételnév;` utasítással. Ez bárhol elhelyezhető, ahol végrehajtható utasítás szerepelhet.
- Néha szükség lehet egy bekövetkezett kivétel újbóli kiváltására.
- Például: egy kivételkezelőben egy adott kivétel lokális kezelése után ugyanazon kivételt továbbadásához.
  - Erre szolgál a RAISE utasítás kivételnév nélküli alakja, amely csak kivételkezelőben alkalmazható.
  - Hatására újra kiváltódik az a kivétel, amely az adott kivételkezelőt aktiválta.

Kollekciótípusok

- Egy kollekció azonos típusú adatelemek egy rendezett együttese.
- A kollekción belül minden elemnek egy egyedi indexe van, amely egyértelműen meghatározza az elem kollekción belüli helyét.

```
kollekció_változó_név(index)
```

- PL/SQL kollekciótípusai:
  - asszociatív tömb
  - beágyazott tábla
  - dinamikus tömb

Kollekciótípusok

- Elemeinek típusa REF CURSOR kivételével tetszőleges PL/SQL típus (akár kollekció is) lehet.
- A 3GL nyelvek tömb fogalmának felelnek meg:
  - Egydimenziós, indexe minden esetben lehet egész (asszociatív tömb esetén sztring is).
  - A többdimenziós tömböket olyan kollekciókkal tudjuk kezelni, melyek elemei maguk is kollekciók.
- A kollekciók átadhatók paraméterként, így segítségükkel adatbázis tábláinak oszlopai mozgathatók az alkalmazások és az adatbázis között.

Kollekciók

- A kollekciók létrehozása két lépcsőben történik:
  - kollekciótípus létrehozása
  - kollekcióváltozó deklarálása:

    ```
    kollekciónév kollekciótípus_név;
    ```
- Egy elemére `kollekciónév(index)` módon hivatkozhatunk.
- Kollekciómetódusok:
  - EXISTS, COUNT, LIMIT, FIRST, LAST, NEXT, PRIOR, EXTEND, TRIM, DELETE

Kollekciómetódusok

- EXISTS – Igaz értéket ad vissza, ha az adott indexű elem létezik a kollekcióban
- COUNT – Visszaadja a kollekció elemeinek számát
- LIMIT – Visszaadja a kollekció maximális méretét
- FIRST, LAST – Visszaadja a kollekció első/utolsó elemének indexét (ha nincs ilyen, akkor NULL-t)
- NEXT, PRIOR – Visszaadja egy megadott indexű elemet követő/megelőző elem indexét (ha nincs ilyen, akkor NULL-t)
- EXTEND – Bővíti a kollekciót
- TRIM – Eltávolítja a kollekció utolsó elemeit
- DELETE – A megadott elemeket törli a kollekcióból

Dinamikus tömb

```
TYPE név IS {VARRAY | VARYING ARRAY} (max_méret)
OF elemtípus [NOT NULL];
```

- A deklarációjakor meg kell adni a maximális méretet, ami rögzíti az indexek felső határát.
- Az adott típusú dinamikus tömb ezután változó számú elemet tartalmazhat, a nulla darabtól a megadott maximális értékig.
- Az elemek folytonosan helyezkednek el és az indexelés 1-től indul.
- A dinamikus tömbhöz adhatóak új elemek, de az elemek száma a maximális méretet nem lépheti túl.
- A dinamikus tömbből törölhetőek elemek, de csak a végéről.

Dinamikus tömb

- Egy ilyen típusú változó tulajdonképpen egy referencia amelynek automatikus kezdőértéke NULL.
- A dinamikus tömböt explicit módon inicializálni példányosítással lehet:
  - A példányosításhoz az adott típus konstruktorát kell meghívni.
  - A konstruktor egy rendszer által létrehozott függvény, amelynek neve megegyezik a típus nevével, paramétereinek száma tetszőleges, paraméterei típusának a kollekciótípus elemeinek típusával kompatibilisnek kell lennie.
  - A dinamikus tömb konstruktorának maximum annyi paraméter adható meg, amennyi a deklarált maximális elemszám.
  - A konstruktor meghívható paraméterek nélkül, ekkor egy üres kollekció jön létre.

Dinamikus tömb

- A dinamikus tömbök elemei lehetnek objektumtípus példányai.
- A dinamikus tömb lehet objektumtípus attribútuma.
- A dinamikus tömb lehet adatbázistábla oszlopának típusa.
- A dinamikus tömb NULL értéke tesztelhető (`IS NULL`).

Dinamikus tömb - kollekciómetódusok

- EXISTS(*i*), COUNT, LIMIT, FIRST, LAST, NEXT(*i*), PRIOR (*i*)
- EXTEND, EXTEND(*n*), EXTEND(*n*,*m*):
  - A paraméter nélküli EXTEND egyetlen NULL elemet helyez el a kollekció végén. Az EXTEND(*n*) *n* darab NULL elemet helyez el a kollekció végén. Az EXTEND(*n*,*m*) esetén az *m* indexű elem *n*-szer elhelyeződik a kollekció végén.
- TRIM, TRIM(*n*):
  - A TRIM eltávolítja a kollekció utolsó elemét. A TRIM(*n*) eltávolítja az utolsó *n* elemet. Ha *n* > COUNT, akkor a SUBSCRIPT_BEYOND_COUNT kivétel váltódik ki.
  - A TRIM a belső méreten operál és az eltávolított elemek helye nem őrződik meg, ezért egy értékadással nem adható nekik új érték.

Dinamikus tömb - kollekciómetódusok

- DELETE:
  - A paraméter nélküli DELETE törli a kollekció összes elemét (egy üres kollekció jön létre).

Dinamikus tömb

- Példa:

```
DECLARE
  TYPE varray_type IS VARRAY(5) OF PLS_INTEGER;
  v1 varray_type;
  v2 varray_type;
  v3 varray_type := varray_type();
  v4 varray_type := varray_type(10,20,30);
BEGIN
    v2 := varray_type(1, 2, 3, 4, 5);
END;
```

Dinamikus tömb - kivételek

- COLLECTION_IS_NULL: NULL értékű kollekcióra metódus meghívása (az EXISTS kivételével)
- SUBSCRIPT_BEYOND_COUNT: az elemszámnál nagyobb indexű elemre hivatkozáskor
- SUBSCRIPT_OUTSIDE_LIMIT: érvényes tartományon kívüli indexhivatkozás esetén (pl. -1)
- VALUE_ERROR: ha az index NULL, vagy nem konvertálható a kulcs típusára