Adatmenedzsment

5. gyakorlat

Asszociatív tömb

```
TYPE név IS TABLE OF elemtípus [NOT NULL]
INDEX BY indextípus;
```

- Kulcs-érték párok halmaza (hash tábla).
- Az indextípus PLS_INTEGER, BINARY_INTEGER, VARCHAR2(n), STRING(n) lehet.
- Az asszociatív tömb esetén az indexeknek sem az alsó, sem a felső határa nem rögzített.
- Az *i* indexű elemnek történő értékadás létrehozza az adott elemet, ha nem létezett, és felülírja az értékét, ha létezett.
- Ha az index NULL, vagy nem konvertálható a kulcs típusára (akár a kulcs típusának korlátozása miatt), akkor a VALUE_ERROR kivétel következik be.

Asszociatív tömb

- Az asszociatív tömb csak a PL/SQL programokban használható, nem lehet adatbázistábla oszlopának típusa, és nem lehet adatbázis objektum.
- Kollekciómetódusok:

  EXISTS(*i*), COUNT, LIMIT, FIRST, LAST, NEXT(*i*), PRIOR(*i*), DELETE, DELETE(*i*), DELETE(*i*,*j*)
- DELETE:
  - A paraméter nélküli DELETE törli a kollekció összes elemét (egy üres kollekció jön létre).
  - DELETE(*i*) törli az *i* indexű elemet, DELETE(*i*,*j*) pedig az *i* és *j* indexek közé eső minden elemet.
  - Ha $i > j$ vagy valamelyik NULL, akkor a DELETE nem csinál semmit.

Asszociatív tömb - kivételek

- NO_DATA_FOUND: nem létező elemre történő hivatkozáskor
- VALUE_ERROR: ha az index NULL, vagy nem konvertálható a kulcs típusára
- Példa:

```
DECLARE
  TYPE t_a IS TABLE OF NUMBER(3) INDEX BY PLS_INTEGER;
  v_a t_a;
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_a(1));
  EXCEPTION WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE(SQLCODE || '  ' || SQLERRM);
END;
```

Beágyazott tábla

```
TYPE név IS TABLE OF elemtípus [NOT NULL];
```

- Indexe egész típusú, aminek alsó határa 1, a felső határ nem rögzített.
- Bármelyik elem törölhető (de a helyén egy „lyuk” keletkezik).
- A beágyazott tábla egy speciális objektum típus.

Beágyazott tábla

- Egy beágyazott tábla deklarálásakor egy referencia jön létre, aminek automatikus kezdőértéke NULL.
  - A beágyazott táblát explicit módon inicializálni az objektumtípusnak megfelelően 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 megfelelő kollekciótípus elemeinek típusával kompatibilisnek kell lennie.
  - A konstruktor meghívható paraméterek nélkül, ekkor egy üres kollekció jön létre.

Beágyazott tábla

- A beágyazott tábla elemei lehetnek objektumtípus példányai.
- A beágyazott tábla lehet objektumtípus attribútuma.
- A beágyazott tábla lehet adatbázistábla oszlopának típusa
- A beágyazott tábla NULL értéke tesztelhető (IS NULL).
- Beágyazott táblák egyenlősége is vizsgálható akkor, ha azonos típusúak és az elemek is összehasonlíthatók egyenlőség szerint.
  - Viszont rendezettségre nézve a beágyazott táblák sem hasonlíthatók össze.

Beágyazott tábla - 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 végére. EXTEND(*n*,*m*) esetén az *m* indexű elem *n*-szer elhelyeződik a kollekció végén.
  - A PL/SQL megtartja a törölt elemek helyét, ezeknek új érték adható.
  - Az EXTEND használatánál, ha a kollekció végén törölt elemek vannak, a bővítés azok után történik.

Beágyazott tábla - kollekciómetódusok

- TRIM , TRIM(n)
  - TRIM eltávolítja a kollekció utolsó elemét, a TRIM(n) pedig az utolsó n elemet. Ha n > COUNT, akkor a SUBSCRIPT_BEYOND_COUNT kivétel váltódik ki.
  - Az eltávolított elemek helye nem őrződik meg, ezért egy értékadással nem adható nekik új érték.

Beágyazott tábla - kollekciómetódusok

- DELETE, DELETE(*i*), DELETE(*i*,*j*)
  - A paraméter nélküli DELETE törli a kollekció összes elemét (egy üres kollekció jön létre helyfoglalással együtt).
  - DELETE(*i*) törli az *i* indexű elemet, a DELETE(*i*,*j*) pedig az *i* és *j* indexek közé eső minden elemet (beleértve *i*-t és *j*-t is). Ha $i > j$ vagy valamelyik NULL, akkor a DELETE nem csinál semmit.

Beágyazott tábla - kollekciómetódusok

- DELETE, DELETE(*i*), DELETE(*i*,*j*)
  - Ha beágyazott táblából paraméteres DELETE metódussal törlünk egy vagy több elemet, az lyukat hozhat létre a beágyazott táblában.
  - Az így törölt elemek által lefoglalt hely továbbra is a memóriában marad (a kollekció belső mérete nem változik), de az elemek értékére történő hivatkozás NO_DATA_FOUND kivételt vált ki.
  - A törölt indexeket a FIRST, LAST, NEXT és PRIOR metódusok figyelmen kívül hagyják, a COUNT értéke a törölt elemek számával csökken.
  - A paraméteres alakkal törölt elemeket újra lehet használni egy értékadás után.

Beágyazott tábla - kivételek

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

Együttes hozzárendelés

- Hozzárendelés: az a tevékenység, amely során egy PL/SQL változónak egy SQL utasításban adunk értéket.
- Együttes hozzárendelés: Egy kollekció minden elemének egyszerre történő hozzárendelése.
- FORALL: PL/SQL-oldali együttes hozzárendelés eszköze
- BULK COLLECT: SQL-oldali együttes hozzárendelés eszköze
- Az együttes hozzárendelés csökkenti a PL/SQL és SQL motorok közötti átváltások számát és így növeli a teljesítményt.

BULK COLLECT

- A SELECT, INSERT, DELETE, UPDATE, FETCH utasítások INTO utasításrészében használható, és az SQL-motortól az együttes hozzárendelést kéri.
- Általános alakja:

```
BULK COLLECT INTO kollekciónév [, kollekciónév]…
```

- A kollekciókat nem kell inicializálni.
- Az adatokat a kollekcióban az 1-es indextől kezdve helyezi el folyamatosan, felülírva a korábbi elemeket.
- Csak szerveroldali programokban alkalmazható.

BULK COLLECT - példa

```
DECLARE
    TYPE T_NT IS TABLE OF EMPLOYEES%ROWTYPE;
    V_NT T_NT;

BEGIN
    SELECT *
    BULK COLLECT INTO V_NT
    FROM EMPLOYEES
    WHERE DEPARTMENT_ID = 50;
    FOR I IN 1 .. V_NT.COUNT
    LOOP
        DBMS_OUTPUT.PUT_LINE(V_NT(I).LAST_NAME);
    END LOOP;

END;
```