Adatbázisban tárolt kollekciók

- Dinamikus tömb és beágyazott tábla lehet

```
CREATE TYPE t_beagy IS TABLE OF NUMBER;
CREATE TYPE t_dint IS VARRAY(5) OF NUMBER;
CREATE TABLE koll_tab (
  azon NUMBER PRIMARY KEY,
  szamok t_beagy,
  lotto t_dint
) NESTED TABLE szamok STORE AS koll_szamok;
INSERT INTO koll_tab VALUES (1, t_beagy(),
  t_dint(23, 47, 52, 53, 88));
```

TABLE operátor

`TABLE(kollekciókifejezés)`

- A kollekció elemeihez, mint egy tábla soraihoz férünk hozzá (kibontás, unnest)

```
SELECT * FROM
         TABLE(SELECT lotto FROM koll_tab);
```

```
insert into table(select szamok
                  from koll_tab
                  where azon=1)
 values (5);

update table(select szamok
                  from koll_tab
                  where azon=1)
set column_value=3
where column_value=5;

delete table(select szamok
                  from koll_tab
                  where azon=1)
where column_value=3;
```

CAST függvény

```
CAST({kifejezés|(alkérdés)|MULTISET(alkérdés)}
  AS típusnév)
```

- Típuskényszerítés
  - kifejezést vagy egy értéket szolgáltató alkérdést adott típusúvá
  - vagy MULTISET esetén akárhány értéket szolgáltató alkérdést kollekció típusúvá alakít

```
UPDATE koll_tab
   SET szamok=CAST(lotto AS t_beagy)
 WHERE azon=1;
```

```
CREATE TYPE T_Dinamikus IS VARRAY(10) OF VARCHAR2(100);
/
CREATE TYPE T_Beagyazott IS TABLE OF varchar2(100);
/
DECLARE
  v_Dinamikus   T_Dinamikus;
  v_Beagyazott  T_Beagyazott;
BEGIN
  SELECT CAST(v_Beagyazott AS T_Dinamikus)
    INTO v_Dinamikus
    FROM dual;

  SELECT CAST(MULTISET(SELECT cim FROM konyv ORDER BY
  UPPER(cim))
              AS T_Beagyazott)
    INTO v_Beagyazott
    FROM dual;
END;
/
```

```
create type t_nt_number_5 is
   table of number(5);
create table nt_elemek
(nev varchar2(50),o_nt t_nt_number_5)
  nested table o_nt store
  as nt_elemek_nt;

insert into nt_elemek
values ('elso',cast(multiset(select
  azon from orszagok where
  foldresz='Európa') as t_nt_number_5))
```

Kollekciók – Beágyazott tábla

- SQL logikai operátorok (*bt1*, *bt2* beágyazott táblák):
  - *bt1*=*bt2*, *bt1*<>*bt2*: két beágyazott tábla megegyezik, ha ugyanaz a típusuk és számosságuk, valamint ugyanazon multihalmazt adják meg
  - *bt1* IN *bt_lista*: *bt1* benne van-e *bt_listá*ban?
  - *bt1* IS [NOT] A SET: *bt1* elemei között van-e ismétlődés?
  - *bt1* IS [NOT] EMPTY: *bt1* üres-e?
  - *e* MEMBER [OF] *bt1*: az *e* elem benne van-e *bt1*-ben?
  - *bt1* SUBMULTISET [OF] *bt2*: *bt1* részmultihalmaza-e *bt2*-nek?

```
declare
  type t_nt is table of number(5);
  nt1 t_nt;
  nt2 t_nt;
begin
nt1:=t_nt(1,2,5);
if nt1 is a set
   then dbms_output.put_line('halmaz');
   else dbms_output.put_line('nem halmaz');
end if;
nt2:=t_nt();
if nt2 is empty
   then dbms_output.put_line('üres');
   else dbms_output.put_line('nem üres');
end if;
nt2:=t_nt(1,1,1,2,2,5,5,3);
if nt1 submultiset of nt2
   then dbms_output.put_line('részhalmaza');
   else dbms_output.put_line('nem részhalmaza');
end if;
end;
```

```
select nev from nt_elemek n
where o_nt is a set;

select nev from nt_elemek n
where o_nt is empty;

select nev from nt_elemek n
where 1 member of o_nt;

select nev from nt_elemek n
where t_nt_number_5(1,2,3) submultiset of o_nt;

select nev from nt_elemek n
where t_nt_number_5(1,2,3,5,6,7) =o_nt;
```

Kollekciók – Beágyazott tábla

- SQL kollekció operátorok (*bt1*, *bt2* beágyazott táblák):
  - *bt1* `MULTISET EXCEPT [{ALL|DISTINCT}]` *bt2:* ***bt1*** és *bt2* közötti multihalmaz-műveletek, különbségképzés
  - *bt1* `MULTISET INTERSECT [{ALL|DISTINCT}]` *bt2:* *bt1* és *bt2* közötti multihalmaz-műveletek, metszet
  - *bt1* `MULTISET UNION [{ALL|DISTINCT}]` *bt2:* ***bt1*** és *bt2* közötti multihalmaz-műveletek, unió

Kollekciók – Beágyazott tábla

- PL/SQL-ben is használható SQL függvények:
  - `CARDINALITY(`*`bt1`*`)`: *bt1* elemszámát adja meg (itt *bt1* dinamikus tömb is lehet!)
  - `SET(`*`bt1`*`)`: eredménye egy beágyazott tábla, amelyet *bt1*-ből az ismétlődések elhagyásával kapunk

```
declare
type t_nt is table of number(5);
nt1 t_nt;
nt2 t_nt;
nt3 t_nt;
procedure bejar(p_nt t_nt) is
 i pls_integer;
 begin
  i:=p_nt.first;
  while i is not null
     loop
     dbms_output.put(p_nt(i));
     i:=p_nt.next(i);
     end loop;
 dbms_output.put_line(null);
 end;
…
```

```
…
begin
 nt1:=t_nt(1,2,3,3,4);
 nt2:=t_nt(1,3,3,3,5,5,7);
 nt3:=nt2 multiset except nt1; bejar(nt3);
 nt3:=nt2 multiset except all nt1; bejar(nt3);
 nt3:=nt2 multiset except distinct nt1; bejar(nt3);
 nt3:=nt2 multiset intersect nt1; bejar(nt3);
 nt3:=nt2 multiset intersect all nt1; bejar(nt3);
 nt3:=nt2 multiset intersect distinct nt1; bejar(nt3);
 nt3:=nt2 multiset union nt1; bejar(nt3);
 nt3:=nt2 multiset union all nt1; bejar(nt3);
 nt3:=nt2 multiset union distinct nt1; bejar(nt3);
 nt3:=set(nt2);bejar(nt3);
 dbms_output.put_line('nt2 elemeinek
   száma:'||cardinality(nt2));
end;
/
```

```
insert into nt_elemek
values ('multiset_elso', t_nt_number_5(2,2,3,3,5,6));
insert into nt_elemek
values ('multiset_masodik',
  t_nt_number_5(1,1,2,2,2,3,3,7));
insert into nt_elemek
values ('multiset_harmadik',(select o_nt
           from nt_elemek where nev='multiset_elso')
         multiset union distinct
              (select o_nt
           from nt_elemek where nev='multiset_masodik'));
select cardinality(o_nt)
from nt_elemek n
where n.nev=‘multiset_elso‘;

select set(o_nt)
from nt_elemek n
where n.nev='multiset_elso'
```

```
CREATE OR REPLACE TYPE list IS TABLE OF NUMBER;
/
CREATE OR REPLACE FUNCTION format_list(set_in LIST) RETURN VARCHAR2 IS
  retval VARCHAR2(2000);
BEGIN
  IF set_in IS NULL THEN
    dbms_output.put_line('Result: <Null>');
  ELSIF set_in.COUNT = 0 THEN
    dbms_output.put_line('Result: <Empty>');
  ELSE
    FOR i IN set_in.FIRST..set_in.LAST LOOP
      IF i = set_in.FIRST THEN
        IF set_in.COUNT = 1 THEN
          retval := '('||set_in(i)||')';
        ELSE
          retval := '('||set_in(i);
        END IF;
      ELSIF i <> set_in.LAST THEN
        retval := retval||', '||set_in(i);
      ELSE
        retval := retval||', '||set_in(i)||')';
      END IF;
    END LOOP;
  END IF;
  RETURN retval;
END format_list;
/
```

```
DECLARE
  a LIST := list(1,2,3,4);
  b LIST := list(4,5,6,7);
BEGIN
  dbms_output.put_line(format_list(a MULTISET EXCEPT b));
END;
/
```

(1, 2, 3)

A PL/SQL eljárás sikeresen befejeződött.

```
DECLARE
  a LIST := list(1,2,3,4);
  b LIST := list(4,5,6,7);
BEGIN
  dbms_output.put_line(format_list(a MULTISET INTERSECT b));
END;
/
```

(4)

A PL/SQL eljárás sikeresen befejeződött.

```
DECLARE
  a LIST := list(1,2,3,4);
  b LIST := list(4,5,6,7);
BEGIN
  dbms_output.put_line(format_list(a MULTISET UNION b));
END;
/
```

(1, 2, 3, 4, 4, 5, 6, 7)

A PL/SQL eljárás sikeresen befejeződött.

```
DECLARE
  a LIST := list(1,2,3,4);
  b LIST := list(4,5,6,7);
BEGIN
  dbms_output.put_line(format_list(a MULTISET UNION DISTINCT b));
END;
/
```

(1, 2, 3, 4, 5, 6, 7)

A PL/SQL eljárás sikeresen befejeződött.

```
DECLARE
  a LIST := list(1,2,3,4);
  b LIST := list(4,5,6,7);
BEGIN
  dbms_output.put_line(format_list(SET(a MULTISET UNION b)));
END;
/
```

(1, 2, 3, 4, 5, 6, 7)

A PL/SQL eljárás sikeresen befejeződött.

```
DECLARE
  a LIST := list(1,2,3,3,4,4,5,6,6,7);
BEGIN
  dbms_output.put_line(format_list(SET(a)));
END;
/
```

(1, 2, 3, 4, 5, 6, 7)

A PL/SQL eljárás sikeresen befejeződött.

```
DECLARE
  a LIST := list(1,2,3,3,4,4,5,6,6,7);
  n PLS_INTEGER := 1;
BEGIN
  IF n MEMBER OF a THEN dbms_output.put_line('Benne van!');
  ELSE                  dbms_output.put_line('Nincs benne!');
  END IF;
END;
/
```

Benne van!

A PL/SQL eljárás sikeresen befejeződött.

```
DECLARE
  a LIST := list(1,2,3,3,4,4,5,6,6,7);
BEGIN
  dbms_output.put_line(CARDINALITY(a));
  dbms_output.put_line(CARDINALITY(SET(a)));
END;
/
```

10
7

A PL/SQL eljárás sikeresen befejeződött.

Kollekciók – Beágyazott tábla

- PL/SQL-ben nem használható SQL függvények:
  - `COLLECT(oszlop)`: a kiválasztott sorokhoz tartozó oszlopértékekből multihalmazt csinál (`CAST` szükséges!)
  - `POWERMULTISET(`*`multihz`*`)`: előállítja a legfeljebb 32 elemű *`multihz`* összes nemüres részmultihalmazának halmazát
  - `POWERMULTISET_BY_CARDINALITY(`*`multihz, db`*`)`: előállítja a legfeljebb 32 elemű *`multihalmaz`* összes *`db`* elemszámú részmultihalmazának halmazát (`db>0`)

```
create type t_nt_nt_number_5 as table of t_nt_number_5;


select cast(powermultiset(t_nt_number_5(1,2,3,5)) as
  t_nt_nt_number_5) from dual;
select *
from table(cast(powermultiset(
  t_nt_number_5(1,2,3,5)) as t_nt_nt_number_5));


select
  cast(POWERMULTISET_BY_CARDINALITY(t_nt_number_5(1,2,3,5)
  ,2) as t_nt_nt_number_5) from dual;
select *
from table(cast(POWERMULTISET_BY_CARDINALITY(
    t_nt_number_5(1,2,3,5),2) as t_nt_nt_number_5));


select cast(collect(azon) as t_nt_number_5)
from orszagok
where foldresz='Európa'
```