Методы сбора, хранения, обработки и анализа данных

Лекция 8

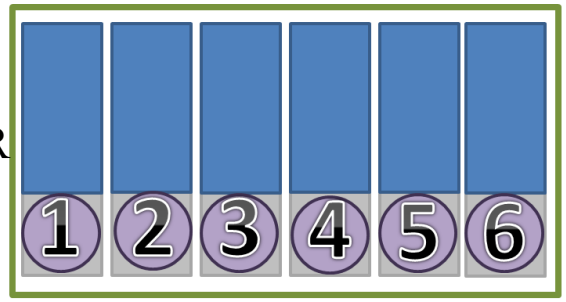
Коллекции

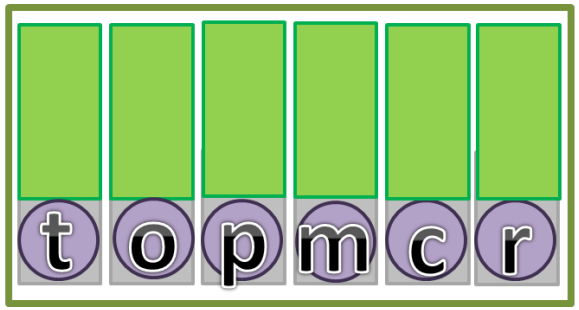
Коллекции

- Тип коллекции
- Экземпляр коллекции
- Однородные элементы
- Одномерность
- Ограниченность
- Плотность/разреженность коллекций
- Индексирование значений
- Внешняя/внутренняя таблица

Коллекции

Index by
PLS_INTEGER

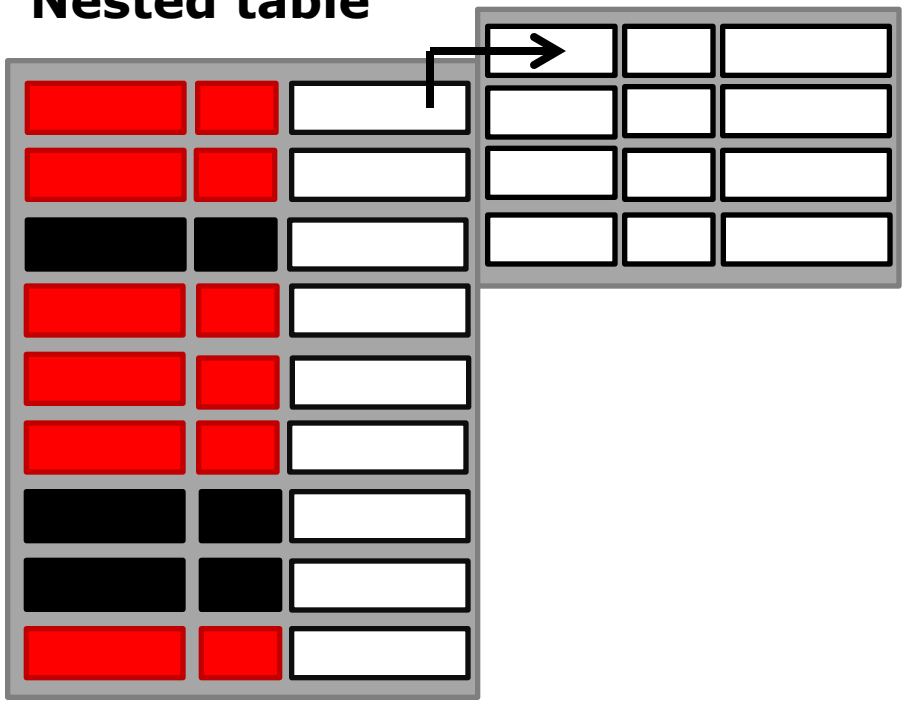




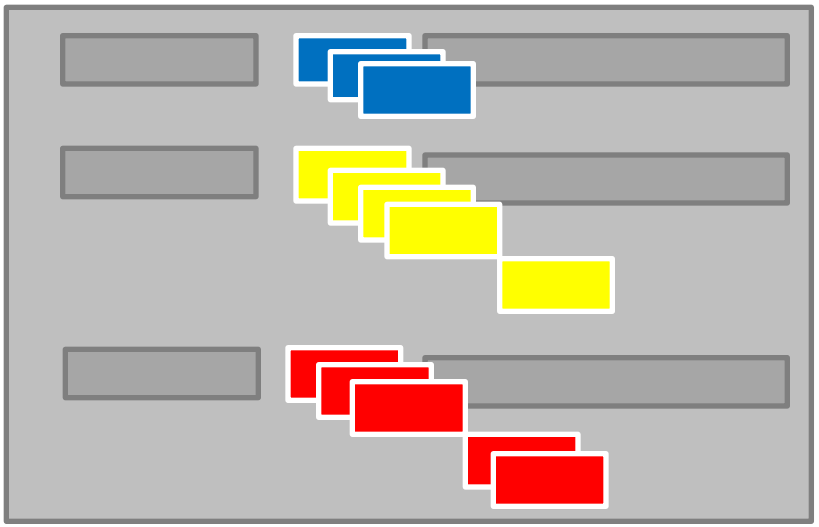
Index by
VARCHAR

Associative array

Nested table



Varray



Термины

- Коллекция состоит из **набора** элементов, причем каждый элемент находится в определенной позиции (имеется **индекс** элемента)
- Необходимо объявить **тип коллекции** – командой TYPE
- Необходимо объявить **коллекцию** – переменную этого типа для дальнейшего использования

Ограниченная и неограниченная коллекция

- Коллекция называется **ограниченной**, если заранее определены границы возможных значений индексов ее элементов, иначе **неограниченной**
- Коллекции типа VARRAY всегда ограничены
- Вложенные таблицы и ассоциативные массивы неограничены

Разреженные и плотные коллекции

- Коллекция называется плотной, если все ее элементы, определены и каждому из них присвоено некоторое значение (таковым может быть и NULL)
- Массивы VARRAY всегда являются плотными
- Вложенные таблицы первоначально всегда плотные, но по мере удаления некоторых элементов становятся разреженными
- Ассоциативные массивы могут быть как разреженными, так и плотными в зависимости от способа их заполнения

Работа с коллекциями

- Объявление коллекций
- Инициализация коллекций
  - Явно с помощью конструктора
  - Неявно при выборке из базы данных
  - Прямым присвоением переменной с другой коллекции такого же типа
- Добавление и удаление элементов
  - Вложенные таблицы и массивы переменной длины – сначала увеличить размер при помощи функции EXTEND, а затем присвоить значения новым элементам

Массивы переменной длины

- Массивы переменной длины – одномерные, связанные коллекции однотипных элементов
- Доступны в рамках PL/SQL и в БД
- Являются плотными

Вложенные таблицы

- Вложенные таблицы – одномерные, несвязанные коллекции однотипных элементов
- Доступны в рамках PL/SQL и как поля таблицы в БД
- Изначально являются плотными, но могут впоследствии становиться разреженными

Пример

- Массивы переменной длины используются для хранения рабочих дней
- Вложенные таблицы используются для хранения месяцев отпуска

```
CREATE TYPE t_work_days IS
            VARRAY(7) OF NUMBER;

CREATE TYPE t_vacation_month IS
            TABLE OF VARCHAR2(15);
```

```
CREATE TABLE employee (
  empno NUMBER(4),
  ename VARCHAR2(25),
  vacation_months t_vacation_month,
  working_days t_work_days)
  NESTED TABLE vacation_months STORE AS
  employee_vacation_months_tab;
```

Пример

```
--- insert data
INSERT INTO employee
    VALUES (12, 'Julia',
            t_vacation_month('June','July','August'),
            t_work_days(1,1,1,1,1,0,0));
INSERT INTO employee
    VALUES (13, 'Sonia',
            t_vacation_month('June','July','August','September'),
            t_work_days(1,1,1,1,1,0,0));
INSERT INTO employee
    VALUES (14, 'Lidya',
            t_vacation_month('May','September'),
            t_work_days(1,1,1,1,1,1,0));
INSERT INTO employee
    VALUES (15, 'Elizabeth',
            t_vacation_month('February','August'),
            t_work_days(1,1,1,1,1,1,0));
INSERT INTO employee
    VALUES (16, 'Margareth',
            t_vacation_month('March','September'),
            t_work_days(1,1,1,1,1,1,0));
COMMIT;
```

Используются конструкторы типов по умолчанию

	EMPNO	ENAME	VACATION_MONTHS	WORKING_DAYS
1	12	Julia	TEST_USER.T_VACATION_MONTH('June', 'July', 'August')	TEST_USER.T_WORK_DAYS(1, 1, 1, 1, 1, 0, 0)
2	13	Sonia	TEST_USER.T_VACATION_MONTH('June', 'July', 'August', 'September')	TEST_USER.T_WORK_DAYS(1, 1, 1, 1, 1, 0, 0)
3	14	Lidya	TEST_USER.T_VACATION_MONTH('May', 'September')	TEST_USER.T_WORK_DAYS(1, 1, 1, 1, 1, 1, 0)
4	15	Elizabeth	TEST_USER.T_VACATION_MONTH('February', 'August')	TEST_USER.T_WORK_DAYS(1, 1, 1, 1, 1, 1, 0)
5	16	Margareth	TEST_USER.T_VACATION_MONTH('March', 'September')	TEST_USER.T_WORK_DAYS(1, 1, 1, 1, 1, 1, 0)

Пример

```
-- in PL/SQL
DECLARE
    vacation_months  t_vacation_month;
    work_days t_work_days;
BEGIN
    vacation_months := t_vacation_month('September');
    work_days := t_work_days(1,1,1,1,1,0,0);
    INSERT INTO employee VALUES (17, 'Mary', vacation_months, work_days);
END;
```

	EMPNO	ENAME	VACATION_MONTHS	WORKING_DAYS
1	12	Julia	TEST USER.T VACATION MONTH('June', 'July', 'August')	TEST USER.T WORK DAYS(1, 1, 1, 1, 1, 0, 0)
2	13	Sonia	TEST USER.T VACATION MONTH('June', 'July', 'August', 'September')	TEST USER.T WORK DAYS(1, 1, 1, 1, 1, 0, 0)
3	14	Lidya	TEST USER.T VACATION MONTH('May', 'September')	TEST USER.T WORK DAYS(1, 1, 1, 1, 1, 1, 0)
4	15	Elizabeth	TEST USER.T VACATION MONTH('February', 'August')	TEST USER.T WORK DAYS(1, 1, 1, 1, 1, 1, 0)
5	16	Margareth	TEST USER.T VACATION MONTH('March', 'September')	TEST USER.T WORK DAYS(1, 1, 1, 1, 1, 1, 0)
6	17	Mary	TEST USER.T VACATION MONTH('September')	TEST USER.T WORK DAYS(1, 1, 1, 1, 1, 0, 0)

Методы коллекций

- count – подсчет количества элементов коллекции
- delete – удаление элементов без сжатия коллекции
- trim – удаление элементов с сжатием коллекции
- extend – расширение коллекции
- first / last – первый /последний индекс коллекции

Пример – count

```
-- count - количество элементов в коллекции
DECLARE
    vacation_months  t_vacation_month := t_vacation_month('June','July','August','September');
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: count ---');
    FOR ind IN 1 .. 4
        LOOP
            dbms_output.put_line(ind|| ' '|| vacation_months(ind));
        END LOOP;
    DBMS_OUTPUT.PUT_LINE('vacation_months.count = '|| vacation_months.count);
END;
```

```
--- Collection methods: count ---
1 June
2 July
3 August
4 September
vacation_months.count = 4
```

Пример – delete, exists

```
-- delete N - удаляет N элемент
-- exists - проверка, существует ли элемент
DECLARE
    vacation_months  t_vacation_month := t_vacation_month('March','May','June','July','August','September');
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: delete ---');
        FOR ind IN 1 .. 6
            LOOP
                dbms_output.put_line(ind|| ' '|| vacation_months(ind));
            END LOOP;
        vacation_months.delete (5);
        FOR ind IN 1 .. 6
            LOOP
                IF vacation_months.exists(ind)
                    THEN dbms_output.put_line(ind|| ' '|| vacation_months(ind));
                    ELSE dbms_output.put_line('No data');
                END IF;
            END LOOP;
END;
```

```
--- Collection methods: delete ---
1 March
2 May
3 June
4 July
5 August
6 September
1 March
2 May
3 June
4 July
No data
6 September
```

Пример – delete

```
-- delete N, M - удаляет от N до M элемента
DECLARE
    vacation_months  t_vacation_month :=
    t_vacation_month('March','May','June','July','August','September');
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: delete ---');
        FOR ind IN 1 .. 6
            LOOP
                dbms_output.put_line(ind|| ' '|| vacation_months(ind));
            END LOOP;
        vacation_months.delete (3, 5);
        DBMS_OUTPUT.PUT_LINE('vacation_months.count = '|| vacation_months.count);
END;
```

```
--- Collection methods: delete ---
1 March
2 May
3 June
4 July
5 August
6 September
vacation_months.count = 3
```

Пример – delete

```
-- delete in varray - only all elements! -- err
DECLARE
    work_days  t_work_days := t_work_days(1,1,1,1,1,1,0);
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: delete ---');
    DBMS_OUTPUT.PUT_LINE('work_days.count = '|| work_days.count);
    work_days.delete (3);
    DBMS_OUTPUT.PUT_LINE('work_days.count = '|| work_days.count);
END;
```

```
Error report -
ORA-06550: Строка 6, столбец 5:
PLS-00306: wrong number or types of arguments in call to 'DELETE'
ORA-06550: Строка 6, столбец 5:
PL/SQL: Statement ignored
06550. 00000 -  "line %s, column %s:\n%s"
*Cause:    Usually a PL/SQL compilation error.
*Action:
```

Пример – delete

```
DECLARE
    work_days  t_work_days := t_work_days(1,1,1,1,1,1,0);
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: delete ---');
    DBMS_OUTPUT.PUT_LINE('work_days.count = '|| work_days.count);
    work_days.delete;
    DBMS_OUTPUT.PUT_LINE('work_days.count = '|| work_days.count);
END;
```

```
--- Collection methods: delete ---
work_days.count = 7
work_days.count = 0
```

Пример – first, last, prior, next

```
-- first, last - возвращают первый и последний индексы элементов коллекции
-- prior, next - возвращают предыдущий и последующий индексы для текущего элемента коллекции
DECLARE
    vacation_months  t_vacation_month :=
    t_vacation_month('March','May','June','July','August','September');
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: first last ---');
    dbms_output.put_line('First = ' || vacation_months.first);
    dbms_output.put_line('Last = ' || vacation_months.last);

    vacation_months.delete(4);

    dbms_output.put_line('First after delete = ' || vacation_months.first);
    dbms_output.put_line('Last after delete = ' || vacation_months.last);

    DBMS_OUTPUT.PUT_LINE('--- Collection methods: prior next ---');

    dbms_output.put_line('Prior of 5 = ' || vacation_months.prior(5));
    dbms_output.put_line('Next of 3 = ' || vacation_months.next(3));
END;
```

```
--- Collection methods: first last ---
First = 1
Last = 6
First after delete = 1
Last after delete = 6
--- Collection methods: prior next ---
Prior of 5 = 3
Next of 3 = 5
```

Пример – trim

```
-- trim - удаляет N элементов коллекции
DECLARE
    vacation_months  t_vacation_month :=
        t_vacation_month('March','May','June','July','August','September');
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: trim ---');
    FOR ind IN 1 .. vacation_months.count
        LOOP
            dbms_output.put_line(ind|| ' '|| vacation_months(ind));
        END LOOP;
    vacation_months.trim(2);
    FOR ind IN 1 .. vacation_months.count
        LOOP
            dbms_output.put_line(ind|| ' '|| vacation_months(ind));
        END LOOP;
END;
```

```
--- Collection methods: trim ---
1 March
2 May
3 June
4 July
5 August
6 September
1 March
2 May
3 June
4 July
```

Пример – extend

```
-- extend - добавляет N элементов коллекции
DECLARE
    vacation_months  t_vacation_month :=
        t_vacation_month('March','May','June','July','August','September');
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: extend ---');
    FOR ind IN 1 .. vacation_months.count
        LOOP
            dbms_output.put_line(ind|| ' '|| vacation_months(ind));
        END LOOP;
    vacation_months.extend(2);
    FOR ind IN 1 .. vacation_months.count
        LOOP
            IF (vacation_months(ind) IS NULL)
                THEN dbms_output.put_line(ind|| ' '|| 'NULL element');
                ELSE dbms_output.put_line(ind|| ' '|| vacation_months(ind));
            END IF;
        END LOOP;
END;
```

```
--- Collection methods: exter
1 March
2 May
3 June
4 July
5 August
6 September
1 March
2 May
3 June
4 July
5 August
6 September
7 NULL element
8 NULL element
```

Пример – extend

```
-- extend (N, i) - добавляет N элементов коллекции
-- с инициализацией их значением элемента i
DECLARE
    vacation_months  t_vacation_month :=
        t_vacation_month('March','May','June','July','August','September');
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: extend ---');
    FOR ind IN 1 .. vacation_months.count
        LOOP
            dbms_output.put_line(ind|| ' '|| vacation_months(ind));
        END LOOP;
    vacation_months.extend(2, 3);
    FOR ind IN 1 .. vacation_months.count
        LOOP
            IF (vacation_months(ind) IS NULL)
                THEN dbms_output.put_line(ind|| ' '|| 'NULL element');
                ELSE dbms_output.put_line(ind|| ' '|| vacation_months(ind));
            END IF;
        END LOOP;
END;
```

```
--- Collection metho
1 March
2 May
3 June
4 July
5 August
6 September
1 March
2 May
3 June
4 July
5 August
6 September
7 June
8 June
```

Пример – limit

```
-- limit максимальное количество элементов
DECLARE
    vacation_months  t_vacation_month := t_vacation_month('March','May');
    work_days  t_work_days := t_work_days(1,1,1,1,1,1,0);
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Collection methods: limit ---');
    dbms_output.put_line('nested table limit = ' || vacation_months.limit); --NULL
    dbms_output.put_line('varray limit = ' || work_days.limit);
END;
```

```
--- Collection methods: limit ---
nested table limit =
varray limit = 7
```

Псевдо функции коллекций

- table – превращение коллекции в табличные данные
- multiset – превращение табличных данных в коллекцию
- cast – превращение одного вида коллекции в другой

Псевдо функции пример

```
SELECT ename, working_days
FROM employee
WHERE ename = 'Julia';

SELECT ename, vacation_months
FROM employee
WHERE ename = 'Julia';
```

	ENAME	WORKING_DAYS
1	Julia	TEST USER.T WORK DAYS(1, 1, 1, 1, 1, 0, 0)

	ENAME	VACATION_MONTHS
1	Julia	TEST USER.T VACATION MONTH('June', 'July', 'August')

Псевдо функции пример

```
-- table - преобразование из коллекции в таблицу
SELECT *
FROM TABLE(SELECT working_days
           FROM employee
           WHERE ename = 'Julia');

SELECT *
FROM TABLE(SELECT vacation_months
           FROM employee
           WHERE ename = 'Julia');
```

	COLUMN_VALUE
1	1
2	1
3	1
4	1
5	1
6	0
7	0

	COLUMN_VALUE
1	June
2	July
3	August

Псевдо функции пример

```
-- cast - преобразование одного вида коллекции в другой
CREATE TYPE t_days IS TABLE OF NUMBER(1); -- t_working_days is varray

SELECT *
FROM TABLE(SELECT CAST(working_days as t_days)
             FROM employee
             WHERE ename = 'Julia');
```

	COLUMN_VALUE
1	1
2	1
3	1
4	1
5	1
6	0
7	0

Псевдо функции пример

```
-- multiset - преобразование из таблицы в коллекцию
CREATE TYPE t_emp IS
            TABLE OF VARCHAR2(15);

SELECT CAST(MULTISET(SELECT ename FROM emp WHERE deptno = 10) AS t_emp)
FROM dual;
```

	CAST(MULTISET(SELECTENAMEFROMEMPWHEREDEPTNO=10)AST_EMP)
1	TEST USER.T EMP('KING', 'CLARK', 'Miller')

Ассоциативные массивы

- Ассоциативные массивы – одномерные, неограниченные (по максимальному количеству элементов при создании) коллекции элементов
- Доступны только в рамках PL/SQL
- Изначально являются разреженными, индекс может принимать непоследовательные значения

Ассоциативные массивы

```
--ассоциативный массив
DECLARE
TYPE t_person IS record (ename VARCHAR2(20), dept number(2));
TYPE it_person IS TABLE OF t_person INDEX BY VARCHAR2(3);
it_emp it_person;
idx VARCHAR2(3);
BEGIN
  dbms_output.put_line('it_emp contains of '||it_emp.count);
  it_emp('DOW').ename := 'John Doe';
  it_emp('DOW').dept := 20;

  it_emp('FLN').ename := 'John Flint';
  it_emp('FLN').dept := 10;

  it_emp('CAE').ename := 'Julius Caesar';
  it_emp('CAE').dept := 30;

  dbms_output.put_line('it_emp contains of '||it_emp.count);
  idx := it_emp.FIRST;
  FOR i IN 1..it_emp.count loop
    dbms_output.put_line(idx||' '||it_emp(idx).ename ||' '||it_emp(idx).dept);
    idx := it_emp.NEXT(idx);
  END loop;
END;
```

```
it_emp contains of 0
it_emp contains of 3
CAE Julius Caesar 30
DOW John Doe 20
FLN John Flint 10
```

Операции multiset

- Операции, позволяющие сравнивать коллекции:
  - Равенство/ неравенство
  - IN
  - UNION, INTERSECT, EXCEPT – объединение, пересечение и разница коллекций, как множеств
  - IS [NOT] EMPTY – пустая коллекция
  - MEMBER OF – входит ли элемент в коллекцию
  - SET OF – наличие дублей
  - SUBMULTISET OF – является ли коллекция подмножеством другой коллекции

Пример multiset

```
-- multiset for nested tables

CREATE TYPE t_emp IS TABLE OF VARCHAR2(10);

CREATE TABLE emp_by_depts (
    deptno NUMBER(2),
    dept_members t_emp)
    NESTED TABLE dept_members
    STORE AS dept_members_table;

CREATE TABLE emp_by_depts_ord (
    deptno NUMBER(2),
    dept_members t_emp)
    NESTED TABLE dept_members
    STORE AS dept_members_table_ord;
```

Пример multiset

```
-- insert all emps by depts in emp_by_depts
INSERT INTO emp_by_depts
SELECT  d.deptno,
        CAST(MULTISET(SELECT ename FROM emp e WHERE e.deptno = d.deptno) AS t_emp)
FROM dept d;
COMMIT;

-- insert all emps by depts in emp_by_depts_ord
INSERT INTO emp_by_depts_ord
SELECT  d.deptno,
        CAST(MULTISET(SELECT e.ename FROM emp e WHERE e.deptno = d.deptno ORDER BY e.ename DESC) AS t_emp)
FROM dept d;
COMMIT;
```

	DEPTNO	DEPT_MEMBERS
1	10	TEST_USER.T_EMP('Miller', 'KING', 'CLARK')
2	20	TEST_USER.T_EMP('SMITH', 'SCOTT', 'JONES', 'FORD', 'ADAMS')
3	30	TEST_USER.T_EMP('WARD', 'TURNER', 'MARTIN', 'JAMES', 'BLAKE', 'ALLEN')
4	40	TEST_USER.T_EMP()

	DEPTNO	DEPT_MEMBERS
1	10	TEST_USER.T_EMP('KING', 'CLARK', 'Miller')
2	20	TEST_USER.T_EMP('JONES', 'FORD', 'SMITH', 'SCOTT', 'ADAMS')
3	30	TEST_USER.T_EMP('BLAKE', 'MARTIN', 'ALLEN', 'TURNER', 'JAMES', 'WARD')
4	40	TEST_USER.T_EMP()

Пример multiset равенство

```sql
-- in SQL
-- compare collections =, !=
select  e.deptno,
        e.dept_members,
        eo.deptno deptno_ord,
        eo.dept_members dept_members_ord
from emp_by_depts e join emp_by_depts_ord eo
on e.dept_members = eo.dept_members;
```

	DEPTNO	DEPT_MEMBERS	DEPTNO_ORD	DEPT_MEMBERS_ORD
1	10	TEST USER.T EMP('KING', 'CLARK', 'Miller')	10	TEST USER.T EMP('Miller', 'KING', 'CLARK')
2	20	TEST USER.T EMP('JONES', 'FORD', 'SMITH', 'SCOTT', 'ADAMS')	20	TEST USER.T EMP('SMITH', 'SCOTT', 'JONES', 'FORD', 'ADAMS')
3	30	TEST USER.T EMP('BLAKE', 'MARTIN', 'ALLEN', 'TURNER', 'JAMES', 'WARD')	30	TEST USER.T EMP('WARD', 'TURNER', 'MARTIN', 'JAMES', 'BLAKE', 'ALLEN')
4	40	TEST USER.T EMP()	40	TEST USER.T EMP()

Пример multiset IN

```
-- in (not in)
select e.deptno, e.dept_members
from emp_by_depts e
where e.dept_members in (
        select eo.dept_members
        from emp_by_depts_ord eo
        where deptno > 20);
```

	DEPTNO	DEPT_MEMBERS
1	30	TEST_USER.T_EMP('BLAKE', 'MARTIN', 'ALLEN', 'TURNER', 'JAMES', 'WARD')
2	40	TEST_USER.T_EMP()

Пример multiset is empty

```
-- is [not] empty;
select e.deptno, e.dept_members
from emp_by_depts e
where e.dept_members is empty;
```

	DEPTNO	DEPT_MEMBERS
1	40	TEST_USER.T_EMP()

Пример multiset submultiset

```
--submultiset of - является ли подмножеством
select e.deptno, e.dept_members, eo.dept_members
from emp_by_depts e, emp_by_depts_ord eo
where e.dept_members SUBMULTISET OF eo.dept_members
and e.deptno = 10;
```

	DEPTNO	DEPT_MEMBERS	DEPT_MEMBERS_1
1	10	TEST_USER.T_EMP('KING', 'CLARK', 'Miller')	TEST_USER.T_EMP('Miller', 'KING', 'CLARK')

Пример multiset is set

```
--set of - содержит ли дубли
select e.deptno, e.dept_members
from emp_by_depts e
where e.dept_members IS NOT A SET; --NULL

UPDATE emp_by_depts e set e.dept_members = t_emp('KING', 'CLARK', 'KING')
WHERE deptno = 10;

select e.deptno, e.dept_members
from emp_by_depts e
where e.dept_members IS NOT A SET; -- KING
```

DEPTNO	DEPT_MEMB...

	DEPTNO	DEPT_MEMBERS
1	10	TEST USER.T EMP('KING', 'CLARK', 'KING')

Пример multiset set ()

```
-- SET(x) - distinct values
select set(e.dept_members)
from emp_by_depts e
where e.dept_members IS NOT A SET;
```

	SET(E.DEPT_MEMBERS)
1	TEST USER.T EMP('KING', 'CLARK')

Пример multiset member of

```
-- member of;
select e.deptno, e.dept_members
from emp_by_depts e
where 'KING' member of e.dept_members;
```

	DEPTNO	DEPT_MEMBERS
1	10	TEST_USER.T_EMP('KING', 'CLARK', 'Miller')

Пример multiset union

```
-- multiset intersect, union, except
-- union -- the same members , different order -- as union all
declare
    v_dept10 t_emp;
    v_dept10_ord t_emp;
    v_union t_emp;
begin
    SELECT dept_members INTO v_dept10
        FROM emp_by_depts WHERE deptno = 10;
    SELECT dept_members INTO v_dept10_ord
    FROM emp_by_depts_ord WHERE deptno = 10;
    v_union := v_dept10 MULTISET UNION v_dept10_ord;
    FOR ind IN 1 .. v_union.COUNT
    LOOP
        dbms_output.put_line (v_union(ind));
    END LOOP;
```

```
KING
CLARK
Miller
Miller
KING
CLARK
```

Пример multiset union

```
-- union -- the same members , the same order -- as union all
declare
    v_dept10 t_emp;
    v_dept10_ord t_emp;
    v_union t_emp;
begin
    SELECT dept_members INTO v_dept10
        FROM emp_by_depts WHERE deptno = 10;
    SELECT dept_members INTO v_dept10_ord
        FROM emp_by_depts WHERE deptno = 10;
    v_union := v_dept10 MULTISET UNION v_dept10_ord;
    FOR ind IN 1 .. v_union.COUNT
    LOOP
        dbms_output.put_line (v_union(ind));
    END LOOP;
end;
```

```
KING
CLARK
Miller
KING
CLARK
Miller
```

Пример multiset union

```
-- union DISTINCT -- the same members , different order -- as union
declare
    v_dept10 t_emp;
    v_dept10_ord t_emp;
    v_union t_emp;
begin
    SELECT dept_members INTO v_dept10
        FROM emp_by_depts WHERE deptno = 10;
    SELECT dept_members INTO v_dept10_ord
    FROM emp_by_depts_ord WHERE deptno = 10;
    v_union := v_dept10 MULTISET UNION DISTINCT v_dept10_ord;
    FOR ind IN 1 .. v_union.COUNT
    LOOP
        dbms_output.put_line (v_union(ind));
    END LOOP;
end;
```

```
KING
CLARK
Miller
```

Пример multiset intersect

```
-- intersect - the same members, different order
declare
    v_dept10 t_emp;
    v_dept10_ord t_emp;
    v_union t_emp;
begin
    SELECT dept_members INTO v_dept10
        FROM emp_by_depts WHERE deptno = 10;
    SELECT dept_members INTO v_dept10_ord
    FROM emp_by_depts_ord WHERE deptno = 10;
    v_union := v_dept10 MULTISET INTERSECT v_dept10_ord;
    FOR ind IN 1 .. v_union.COUNT
    LOOP
        dbms_output.put_line (v_union(ind));
    END LOOP;
end;
```

```
KING
CLARK
Miller
```

Пример multiset except

```
-- except - the same members, different order
declare
    v_dept10 t_emp;
    v_dept10_ord t_emp;
    v_union t_emp;
begin
    SELECT dept_members INTO v_dept10
        FROM emp_by_depts WHERE deptno = 10;
    SELECT dept_members INTO v_dept10_ord
        FROM emp_by_depts_ord WHERE deptno = 10;
    v_union := v_dept10 MULTISET EXCEPT v_dept10_ord;
    dbms_output.put_line ('Count of element in result collection = ' || v_union.count);
end;
```

```
Count of element in result collection = 0
```

Использование коллекций

- Как поле таблицы
- Как параметр процедуры/функции
- Как возвращаемое функцией значение
- Как компонент записи
- Как атрибут объектного типа

Пример – как поле таблицы

```
-- 1. Как поле таблицы
CREATE TYPE t_person AS OBJECT
 (p_name VARCHAR2(20), p_dob DATE);

CREATE TYPE t_children IS
            TABLE OF t_person;

CREATE TABLE employee (
        empno NUMBER(4),
        ename VARCHAR2(15),
        children t_children)
NESTED TABLE children STORE AS
  emp_children_tab;
```

Пример – как параметр процедуры

```
-- 2. Как параметр процедуры/функции
CREATE OR REPLACE PROCEDURE new_emp ( p_empno_in employee.empno%TYPE,
                            p_ename_in employee.ename%TYPE,
                            p_children_in t_children)
AS
BEGIN
    INSERT INTO employee VALUES (p_empno_in, p_ename_in, p_children_in);
END;

-- exec
declare
    v_empno employee.empno%type := '7788';
    v_ename employee.ename%type := 'John Dow';
    v_children t_children :=
        t_children (t_person('Eva','01/02/2012'),
                    t_person('Adam','01/06/2015'));
begin
    new_emp(v_empno, v_ename, v_children);
end;
```

Пример – как возвращаемое значение

```
-- 3. Как возвращаемое функцией значение
CREATE OR REPLACE FUNCTION emp_chilren ( p_empno_in employee.empno%TYPE)
RETURN t_children
AS
    v_children t_children;
BEGIN
    SELECT children INTO v_children
    FROM employee
    WHERE empno = p_empno_in;
    RETURN v_children;
END;

--
SELECT emp_chilren(7788) FROM dual;
```

```
EMP_CHILREN(7788)(P_NAME, P_DOB)
--------------------------------------------------------------------------
T_CHILDREN(T_PERSON('Eva', '01.02.12'), T_PERSON('Adam', '01.06.15'))
```

Пример – как возвращаемое значение

```
-- 4. Как компонент записи
DECLARE
    TYPE t_emp
    IS RECORD (p_empno employee.empno%TYPE,
               p_ename employee.ename%TYPE,
               p_children t_children);
    v_emp t_emp;
BEGIN
    SELECT *
        INTO v_emp
        FROM employee
        WHERE empno = 7788;
    dbms_output.put_line('Children of employee #'|| v_emp.p_empno || ' '||v_emp.p_ename || ' are:');
    FOR idx IN 1.. v_emp.p_children.COUNT
        LOOP
            dbms_output.put_line(v_emp.p_children(idx).p_name || ', born: ' || v_emp.p_children(idx).p_dob);
        END LOOP;
END;
```

```
Children of employee #7788 John Dow are:
Eva, born: 01.02.12
Adam, born: 01.06.15
```

Пример – как атрибут объектного типа

```
-- 5. Как атрибут объектного типа
CREATE TYPE t_emp AS OBJECT (
 empno NUMBER,
 ename VARCHAR2(15),
 dob DATE,
 e_children t_children,
 MEMBER FUNCTION how_many_children RETURN NUMBER);

CREATE TYPE BODY t_emp AS
    MEMBER FUNCTION how_many_children RETURN NUMBER IS
     BEGIN
        RETURN SELF.e_children.count;
     END;
END;

CREATE TABLE ot_emp OF t_emp
NESTED TABLE e_children STORE AS
  ot_emp_children_tab;
```

Пример – как атрибут объектного типа

```
CREATE TABLE ot_emp OF t_emp
NESTED TABLE e_children STORE AS
  ot_emp_children_tab;

DECLARE
    v_empno NUMBER(4) := '7788';
    v_ename VARCHAR2(15) := 'John Dow';
    v_dob DATE := TO_DATE('16/12/1987', 'dd/mm/yyyy');
    v_children t_children :=
        t_children (t_person('Eva','01/02/2012'),
                    t_person('Adam','01/06/2015'));
    v_emp t_emp := t_emp(v_empno, v_ename, v_dob, v_children);
BEGIN
    INSERT INTO ot_emp VALUES v_emp;
    COMMIT;
END;
```

```
SELECT o.empno,
       o.ename,
       o.dob,
       o.how_many_children() hm_children
FROM ot_emp o;
```

	EMPNO	ENAME	DOB	HM_CHILDREN
1	7788	John Dow	16.12.87	2

Сравнение характеристик коллекций

- Размерность?
- Можно ли использовать как поле в таблице?
- Неинициализированное состояние?
- Инициализация?
- Диапазон индексов?
- Разреженность?
- Ограничен по максимальному количеству элементов?
- Можно ли присваивать значение любому элементу?
- Метод расширения и уменьшения?
- Можно ли сравнивать на равенство весь объект целиком?
- Элементы сохраняют позицию при записи или чтении из БД?

Применение коллекций

- Если необходим разреженный массив данных – использовать ассоциативный массив или вложенную таблицу, явно распределяя и удаляя элементы.
- Если индекс элемента может быть отрицательным, то использовать ассоциативный массив.
- Если необходимо задать ограничение на количество элементов коллекции, то используется VARRAY.
- Если необходимо хранить большие объемы данных в коллекции, то используется вложенная таблица.
- Если необходимо сохранять порядок элементов коллекции, то используется VARRAY.

Переключение контекста

- FORALL – цикл, в котором коллекция обрабатывается без переключения контекста, могут быть использованы курсорные атрибуты + SQL%BULK_ROWCOUNT
- BULK COLLECT INTO – помещение данных в коллекцию без переключения контекста

Пример – цикл FOR

```
CREATE TABLE MONTHS(mon VARCHAR2(25) PRIMARY KEY);

-- обычный цикл с переключением контекста
DECLARE
  TYPE t_month_list IS TABLE OF VARCHAR2(25);
  month_list t_month_list :=
  t_month_list('January','April','May','June','July','September', 'December');
BEGIN
  FOR indx IN 1..month_list.count
    LOOP
        INSERT INTO months (mon) VALUES (month_list(indx));
    END LOOP;
END;

SELECT * FROM months;
TRUNCATE TABLE months;
```

	MON
1	April
2	December
3	January
4	July
5	June
6	May
7	September

Пример – цикл FORALL

```
-- цикл FORALL без переключения контекста
DECLARE
  TYPE t_month_list IS TABLE OF VARCHAR2(25);
  month_list t_month_list :=
  t_month_list('January','April','May','June','July','September', 'December');
BEGIN
  FORALL indx IN month_list.first .. month_list.last
        INSERT INTO months (mon) VALUES (month_list(indx));
END;

SELECT * FROM months;
TRUNCATE TABLE months;
```

	MON
1	April
2	December
3	January
4	July
5	June
6	May
7	September

Пример – цикл FOR дубли

```
-- Обработка дублей?
DECLARE
  TYPE t_month_list IS TABLE OF VARCHAR2(25);
  month_list t_month_list :=
  t_month_list('January','February','March','April','May',
  'June','July','November','February','April',
  'May','June','July','September', 'December');
BEGIN
  FOR indx IN 1 .. month_list.count
    LOOP
        INSERT INTO months (mon) VALUES (month_list(indx));
    END LOOP;
EXCEPTION
    WHEN others THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred during iteration ');
        DBMS_OUTPUT.PUT_LINE(sqlerrm);
END;
```

```
Error occurred during iteration
ORA-00001: нарушено ограничение уникальности (TEST_USER.SYS_C0010479)
```

Пример – цикл FORALL дубли

```
-- Обработка ошибок в конструкции save exceptions позволяет
-- добавить данные, а потом обработать ошибки
DECLARE
  TYPE t_month_list IS TABLE OF VARCHAR2(25);
  month_list t_month_list :=
  t_month_list('January','February','March','April','May',
  'June','July','November','February','April',
  'May','June','July','September', 'December');
  error_count NUMBER;
BEGIN
  FORALL indx IN month_list.first .. month_list.last
    SAVE EXCEPTIONS
        INSERT INTO months (mon) VALUES (month_list(indx));
EXCEPTION
    WHEN others
    THEN
      error_count := SQL%BULK_EXCEPTIONS.count;
      dbms_output.put_line('Number of errors is '||error_count);
      FOR indx IN 1..error_count
      LOOP
        dbms_output.put_line('Error ' || indx || '
             occurred during iteration ' ||
              SQL%BULK_EXCEPTIONS(indx).error_index);
        dbms_output.put_line('Error is ' ||
          SQLERRM(-SQL%BULK_EXCEPTIONS(indx).error_code));
      END LOOP;
END;
```

	MON
1	April
2	December
3	February
4	January
5	July
6	June
7	March
8	May
9	November
10	September

```
Number of errors is 13
Error 1
             occurred during iteration 1
Error is ORA-00001: нарушено ограничение уникальности (.)
Error 2
             occurred during iteration 2
Error is ORA-00001: нарушено ограничение уникальности (.)
Error 3
             occurred during iteration 3
Error is ORA-00001: нарушено ограничение уникальности (.)
```

Пример – цикл FORALL дубли

```
DECLARE
  TYPE t_month_list IS TABLE OF VARCHAR2(25);
  month_list t_month_list :=
  t_month_list('January','February','March','April','May',
  'June','July','November','February','April',
  'May','June','July','September', 'December');
  error_count NUMBER;
BEGIN
  FORALL indx IN month_list.first .. month_list.last
    --SAVE EXCEPTIONS
        INSERT INTO months (mon) VALUES (month_list(indx));
EXCEPTION
    WHEN others
    THEN
      error_count := SQL%BULK_EXCEPTIONS.count;
      dbms_output.put_line('Number of errors is '||error_count);
      FOR indx IN 1..error_count
      LOOP
        dbms_output.put_line('Error ' || indx || '
             occurred during iteration ' ||
              SQL%BULK_EXCEPTIONS(indx).error_index);
        dbms_output.put_line('Error is ' ||
          SQLERRM(-SQL%BULK_EXCEPTIONS(indx).error_code));
      END LOOP;
END;
```

	MON
1	April
2	February
3	January
4	July
5	June
6	March
7	May
8	November

```
Number of errors is 1
Error 1
             occurred during iteration 9
Error is ORA-00001: нарушено ограничение уни
```

Пример – FORALL курсорные атрибуты

```
-- Курсорные атрибуты
-- SQL%BULK_ROWCOUNT
DECLARE
   TYPE mnfr_list IS TABLE OF VARCHAR2(3);
   v_mnfr_list mnfr_list
   := mnfr_list ('ACI', 'BIC', 'FEA', 'QSA', 'IMM');
BEGIN
   FORALL indx IN
          v_mnfr_list.first..v_mnfr_list.last
      UPDATE products
         SET QTY_ON_HAND = QTY_ON_HAND / 2
      WHERE MFR_ID = v_mnfr_list (indx);
  DBMS_OUTPUT.PUT_LINE ('Some lines are updated: '||SQL%ROWCOUNT);
  IF SQL%found
    THEN dbms_output.put_line ('Found for update ');
  END IF;

  IF SQL%notfound
    THEN dbms_output.put_line ('Not Found for update ');
  END IF;
  FOR indx IN v_mnfr_list.first .. v_mnfr_list.last
    LOOP
      dbms_output.put_line ( indx|| '  ' || SQL%bulk_rowcount(indx) ) ;
    END LOOP;
END;
```

```
Some lines are updated: 21
Found for update
1   7
2   3
3   2
4   3
5   6
```

FORALL разреженные массивы

```
-- Работа с разреженными коллекциями
-- стандартное обращение - ошибка при появлении разреженности
DECLARE
  TYPE t_month_list IS TABLE OF VARCHAR2(25);
  month_list t_month_list :=
  t_month_list('January','April','May','June','July','September', 'December');
BEGIN
    month_list.delete(4);
    FORALL indx IN month_list.first .. month_list.last
        INSERT INTO months (mon) VALUES (month_list(indx));
END;

SELECT * FROM months;
```

```
Error report -
ORA-22160: элемент с индексом [4] не существует
ORA-06512: на  line 7
22160. 00000 -  "element at index [%s] does not exist"
*Cause:    Collection element at the given index does not exist.
*Action:   Specify the index of an element which exists.
```

MON

FORALL IN INDICES OF

```
-- IN INDICES OF
DECLARE
  TYPE t_month_list IS TABLE OF VARCHAR2(25)
    INDEX BY PLS_INTEGER;
  month_list t_month_list;
  TYPE t_checked_months IS TABLE OF BOOLEAN
    INDEX BY PLS_INTEGER;
  checked_months t_checked_months;

BEGIN
  month_list(1) := 'January';
  month_list(4) := 'April';

  month_list(6) := 'June';
  month_list(7) := 'July';

  month_list(9) := 'September';
  month_list(12) := 'December';

  checked_months(6) := TRUE;
  checked_months(7) := TRUE;

  FORALL indx IN INDICES OF checked_months
        INSERT INTO months(mon) VALUES (month_list(indx));
END;

SELECT * FROM months;
TRUNCATE TABLE months;
```

	MON
1	July
2	June

FORALL IN VALUES OF

```
-- IN VALUES OF
DECLARE
  TYPE t_month_list IS TABLE OF VARCHAR2(25)
    INDEX BY PLS_INTEGER;
  month_list t_month_list;
  TYPE t_checked_months IS TABLE OF PLS_INTEGER
    INDEX BY PLS_INTEGER;
  checked_months t_checked_months;

BEGIN
  month_list(-1) := 'January';
  month_list(-4) := 'April';

  month_list(-6) := 'June';
  month_list(-7) := 'July';

  month_list(-9) := 'September';
  month_list(-12) := 'December';

  checked_months(1) := -6;
  checked_months(2) := -12;

  FORALL indx IN VALUES OF checked_months
        INSERT INTO months(mon) VALUES (month_list(indx));
END;
```

	MON
1	December
2	June

BULK COLLECT INTO

- BULK COLLECT INTO – помещение данных в коллекцию без переключения контекста:
  - Из таблицы
  - Из курсора
  - В одну или несколько коллекций
  - Порциями
  - Возврат измененных данных в коллекцию при DML операциях

Пример – BULK INTO

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7839	KING	PRESIDENT	(null)	17.11.81	400	(null)	10
2	7698	BLAKE	MANAGER	7839	01.05.81	2850	(null)	30
3	7782	CLARK	MANAGER	7839	09.06.81	2450	(null)	10
4	7566	JONES	MANAGER	7839	02.04.81	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	28.09.81	1250	400	30
6	7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
7	7844	TURNER	SALESMAN	7698	08.09.81	1500	0	30
8	7900	JAMES	CLERK	7698	03.12.81	950	(null)	30
9	7521	WARD	SALESMAN	7698	22.02.81	1250	500	30
10	7902	FORD	ANALYST	7566	03.12.81	3000	(null)	20
11	7369	SMITH	CLERK	7902	17.12.80	800	(null)	20
12	7788	SCOTT	ANALYST	7566	09.12.82	3000	(null)	20
13	7876	ADAMS	CLERK	7788	12.01.83	1100	(null)	20
14	7934	Miller	CLERK	7782	23.01.82	1300	(null)	10

```
-- BULK COLLECT множественная запись в коллекции
DECLARE
  TYPE emp_type IS TABLE OF emp%rowtype;
   v_emps emp_type;
BEGIN
   SELECT * BULK COLLECT INTO v_emps FROM emp;
   FOR indx IN 1 ..v_emps.COUNT
   LOOP
        dbms_output.put_line(indx || '  '||
            v_emps(indx).ename|| ' ' ||v_emps(indx).JOB);
   END LOOP;
END;
```

```
1  KING PRESIDENT
2  BLAKE MANAGER
3  CLARK MANAGER
4  JONES MANAGER
5  MARTIN SALESMAN
6  ALLEN SALESMAN
7  TURNER SALESMAN
8  JAMES CLERK
9  WARD SALESMAN
10  FORD ANALYST
11  SMITH CLERK
12  SCOTT ANALYST
13  ADAMS CLERK
14  Miller CLERK
```

Пример – BULK INTO

```
-- BULK COLLECT - несколько коллекций одновременно
DECLARE
  TYPE emp_type IS TABLE OF emp.ename%type;
   v_emps emp_type;
  TYPE dept_type IS TABLE OF emp.deptno%type;
   v_depts dept_type;
BEGIN
   SELECT e.ename, e.deptno BULK COLLECT
    INTO v_emps, v_depts FROM emp e;
   FOR indx IN 1 ..v_emps.COUNT
   LOOP
        dbms_output.put_line(indx || '  '||
            v_emps(indx)|| ' ' ||v_depts(indx));
   END LOOP;
END;
```

```
1  KING 10
2  BLAKE 30
3  CLARK 10
4  JONES 20
5  MARTIN 30
6  ALLEN 30
7  TURNER 30
8  JAMES 30
9  WARD 30
10  FORD 20
11  SMITH 20
12  SCOTT 20
13  ADAMS 20
14  Miller 10
```

Пример – BULK INTO LIMIT

```
-- BULK COLLECT - CURSOR - LIMIT
DECLARE
    CURSOR all_rows_from_emp IS
        SELECT * FROM emp;
    TYPE emp_type IS TABLE OF emp%rowtype;
    v_emps emp_type;
    l_row PLS_INTEGER;
BEGIN
    OPEN all_rows_from_emp;
    LOOP
        FETCH all_rows_from_emp BULK COLLECT
            INTO v_emps LIMIT 5;
        EXIT WHEN  v_emps.count = 0;
        -- Manage every portion of data
        DBMS_OUTPUT.PUT_LINE('--------------------');
        l_row := v_emps.first;
        FOR indx IN 1 ..v_emps.COUNT
            LOOP
                dbms_output.put_line(indx || '  '||
            v_emps(indx).ename|| ' ' ||v_emps(indx).JOB);
            END LOOP;
    END LOOP;
END;
```

```
--------------------
1  KING PRESIDENT
2  BLAKE MANAGER
3  CLARK MANAGER
4  JONES MANAGER
5  MARTIN SALESMAN
--------------------
1  ALLEN SALESMAN
2  TURNER SALESMAN
3  JAMES CLERK
4  WARD SALESMAN
5  FORD ANALYST
--------------------
1  SMITH CLERK
2  SCOTT ANALYST
3  ADAMS CLERK
4  Miller CLERK
```

Пример – RETURNING BULK COLLECT

```
-- RETURNING BULK COLLECT
DECLARE
    TYPE t_lucky_ones IS TABLE OF varchar2(15);
    v_lucky_ones t_lucky_ones :=
        t_lucky_ones ('KING', 'CLARK', 'JONES', 'BLAKE');

    TYPE emp_type IS TABLE OF emp%rowtype;
    v_emps emp_type;
BEGIN
    FORALL  indx in v_lucky_ones .first .. v_lucky_ones .last
    UPDATE emp e SET e.comm = e.sal*0.25 WHERE e.ename = v_lucky_ones(indx)
        RETURNING empno, ename, job, mgr, hiredate, sal, comm, deptno
        BULK COLLECT
        INTO v_emps;
    FOR indx IN 1 ..v_emps.COUNT
    LOOP
        dbms_output.put_line(indx || '  '||v_emps(indx).ename|| ' ' ||
                        v_emps(indx).sal || ' ' ||
                        v_emps(indx).comm);
    END LOOP;
END;
```

```
1  KING 400 100
2  CLARK 2450 612,5
3  JONES 2975 743,75
4  BLAKE 2850 712,5
```

Вопросы?