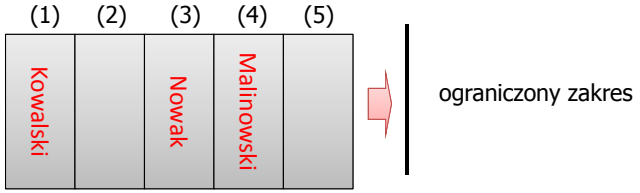
Rozszerzenie obiektowe w SZBD Oracle

Cześć 2. Kolekcje

Kolekcje

- Zbiory obiektów, rodzaje:
  - tablica o zmiennym rozmiarze (ang. *varray*)



  - tablica zagnieżdżona (ang. *nested table*)

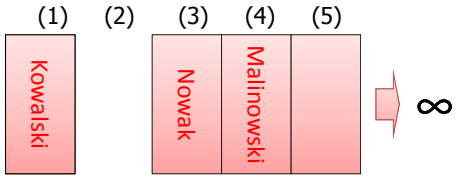


Tabela o zmiennym rozmiarze (1)

- Definiowana jako obiekt schematu lub element programu PL/SQL,
  - przy dostępie z poziomu SQL brak możliwości manipulacji pojedynczymi elementami.
- Przy deklaracji konieczne określenie maksymalnego rozmiaru,
  - późniejsza zmiana rozmiaru:

```
ALTER TYPE nazwa_kolekcji
MODIFY LIMIT nowy_rozmiar
…;
```

- Składnia deklaracji:

```
CREATE TYPE nazwa IS|AS
VARRAY(rozmiar_kolekcji)
OF typ_elementu;
```

```
TYPE nazwa IS|AS
VARRAY(rozmiar_kolekcji)
OF typ_elementu;
```


Tabela o zmiennym rozmiarze (2)

- Zachowuje uporządkowanie elementów.
- Odwołanie do elementu przez liczbowy indeks,
  - jest strukturą gęstą – posiada tyle elementów ile zadeklarowano, elementy bez wartości są puste.
- Składowanie kolekcji będącej elementem obiektu schematu:
  - rozmiar do ok. 4000B – składowanie bezpośrednio w kolumnie,
  - powyżej 4000B – składowane jako LOB,
    - możliwe składowanie pierwszych 4000B danych bezpośrednio w kolumnie.

Tabela o zmiennym rozmiarze (3)

```
CREATE TYPE Telefony AS VARRAY(10) OF VARCHAR(15);

CREATE TABLE znajomi (imie VARCHAR(100), kontakt Telefony);

INSERT INTO znajomi
VALUES ('Bolek', Telefony('1234567'));

INSERT INTO znajomi
VALUES ('Lolek', Telefony('1111111', '2222222'));
```

```
DECLARE
  TYPE Telefony IS VARRAY(10) OF VARCHAR(15);
  moi_znajomi Telefony := Telefony('111111', '222222');
BEGIN
  ...
```


Tabela zagnieżdżona (1)

- Definiowana jako obiekt schematu lub element programu PL/SQL,
  - w obu przypadkach możliwe operowanie na pojedynczych elementach kolekcji.
- Brak wskazania maksymalnego rozmiaru.
- Składnia deklaracji:

```
CREATE TYPE nazwa IS|AS
TABLE OF typ_elementu;
```

```
TYPE nazwa IS|AS
TABLE OF typ_elementu;
```


Tabela zagnieżdżona (2)

- Nie zachowuje uporządkowania elementów.
- Odwołanie do elementu przez liczbowy indeks,
  - jest strukturą rzadką – niezainicjalizowany element nie istnieje, mogą wystąpić przerwy między elementami.
- Składowanie kolekcji będącej elementem obiektu schematu:
  - w osobnej tabeli schematu (out-line),
  - tabela zawiera identyfikator tabeli nadrzędnej,
  - możliwe wskazanie nazwy tabeli i przestrzeni tabel do składowania kolekcji:

```
… NESTED TABLE nazwa_kolekcji STORE AS nazwa_tabeli
[(TABLESPACE nazwa_przestrzeni_tabel)]
```


Tabela zagnieżdżona (3)

```
CREATE TYPE Zespol AS TABLE OF Pracownik;

CREATE TABLE projekty(nazwa VARCHAR(100), wykonawcy Zespol)
NESTED TABLE wykonawcy STORE AS wykonawcy_projektu;

INSERT INTO projekty
VALUES ('Z1',Zespol(Pracownik('Bolek'), Pracownik('Lolek')));

INSERT INTO projekty
VALUES ('Z2',Zespol(Pracownik('Tola')));
```

```
DECLARE
  TYPE Zespol IS TABLE OF Pracownik;
  bdanych Zespol := Zespol(Pracownik('Bolek'),Pracownik('Tola'));
BEGIN
  ...
```

Porównanie typów kolekcji

własność	VARRAY	NESTED TABLE
maksymalny rozmiar	tak	nie
usuwanie elementów ze środka kolekcji	nie	tak
składowanie w bazie danych	in-line/out-line	out-line
zachowanie fizycznego porządku	tak	nie
manipulowanie na pojedynczych elementach kolekcji w SQL	nie	tak

Zapytania do kolekcji (1)

- Zagnieżdżenie kolekcji w zbiorze wynikowym:

```
SELECT imie, kontakt FROM znajomi;

IMIE      KONTAKT
--------- ------------------------------------
Bolek     TELEFONY('1234567')
Lolek     TELEFONY('1111111','2222222')
```

```
SELECT nazwa, wykonawcy FROM projekty;

NAZWA  WYKONAWCY
------ ------------------------------------------------------------------------
Z1     ZESPOL(PRACOWNIK('Bolek',1000,NULL,NULL),PRACOWNIK('Lolek',1000,NULL,NULL))
Z2     ZESPOL(PRACOWNIK('Tola',1000,NULL,NULL))
```


Zapytania do kolekcji (2)

- Usunięcie zagnieżdżenia – klauzula **TABLE**:

```
SELECT imie, k.* FROM znajomi z, TABLE (z.kontakt) k

IMIE       COLUMN_VALUE
---------- ---------------
Bolek      1234567
Lolek      1111111
Lolek      2222222

SELECT COLUMN_VALUE AS numer
FROM TABLE(SELECT kontakt FROM znajomi WHERE imie = 'Lolek');

NUMER
---------------
1111111
2222222
```


Zapytania do kolekcji (3)

- Usunięcie zagnieżdżenia – klauzula **TABLE** (cd):

```
SELECT w.nazwisko FROM projekty, TABLE(wykonawcy) w
WHERE nazwa = 'Z1';

NAZWISKO
----------
Bolek
Lolek

SELECT nazwisko, pensja
FROM TABLE(SELECT wykonawcy FROM projekty WHERE nazwa = 'Z2');

NAZWISKO    PENSJA
---------- ------
Tola          1000
```

Funkcja CAST

- Przekształca wartość typu wyjściowego do wartości zadanego typu.
- Przykład dla typu wbudowanego:

```
SELECT CAST(placa_pod AS VARCHAR2(10)) FROM pracownicy;
```

- Przykład dla kolekcji:

```
SELECT nazwa AS zespol,
       CAST(MULTISET(SELECT nazwisko,placa_pod,etat,null FROM pracownicy
                     WHERE id_zesp = z.id_zesp) AS zespol) AS pracownicy
FROM zespoly z ORDER BY nazwa;

ZESPOL             PRACOWNICY
------------------ ------------------------------------------------------------
ADMINISTRACJA      ZESPOL(PRACOWNIK('WEGLARZ',1730,'DYREKTOR',NULL),
                          PRACOWNIK('MAREK',410.2,'SEKRETARKA',NULL))
ALGORYTMY          ZESPOL(PRACOWNIK('BLAZEWICZ',1350,'PROFESOR',NULL))
BADANIA OPERACYJNE ZESPOL()
```

Operacje zbiorowe na kolekcjach (1)

- Dostępne tylko dla tabel zagnieżdżonych.
- **CARDINALITY** – funkcja zwracająca liczbę elementów kolekcji.

```
SELECT nazwa, CARDINALITY(wykonawcy) AS l_wyk
FROM projekty;

NAZWA     L_WYK
--------- -----
Z1            2
Z2            1
```


Operacje zbiorowe na kolekcjach (2)

- **COLLECT** – funkcja tworząca wielozbiór z danego zbioru elementów,
  - wymaga użycia funkcji CAST.

```
SELECT CAST(COLLECT(Pracownik(nazwisko)) AS Zespol) AS zesp_10
FROM pracownicy WHERE id_zesp = 10;

ZESP_10
--------------------------------------------------------------------------------
ZESPOL(PRACOWNIK('WEGLARZ',1000,NULL,NULL),PRACOWNIK('MAREK',1000,NULL,NULL))
```

- **SET** – tworzy zbiór z kolekcji, zwraca kolekcję po eliminacji duplikatów.

Operacje zbiorowe na kolekcjach (3)

- Operatory wielozbiorowe:
  - działają na dwóch kolekcjach tego samego typu, zwracają nową kolekcję,
  - dostępne operatory:
    - **MULTISET EXCEPT** – różnica kolekcji,
    - **MULTISET INTERSECT** – część wspólna kolekcji,
    - **MULTISET UNION** – suma kolekcji,
  - możliwe dodanie klauzuli **DISTINCT** (np., MULTISET EXCEPT DISTINCT), domyślna klauzula **ALL**.

Operacje zbiorowe na kolekcjach (4)

- Operatory zbiorowe (cd):
  – przykład:

```
SELECT nazwisko FROM TABLE(
    SELECT p1.wykonawcy MULTISET UNION p2.wykonawcy
    FROM projekty p1, projekty p2
    WHERE p1.nazwa = 'Z1' AND p2.nazwa = 'Z2');

NAZWISKO
-------------
BOLEK
LOLEK
TOLA
```


Operacje DML na kolekcji (1)

- Dla tabeli o zmiennym rozmiarze – tylko operacje na całej kolekcji.

```
SELECT kontakt FROM znajomi WHERE imie = 'Bolek';

KONTAKT
------------------------------
TELEFONY('1234567')

UPDATE znajomi
SET kontakt = Telefony('1234567','3333333') WHERE imie = 'Bolek';

SELECT kontakt FROM znajomi WHERE imie = 'Bolek';

KONTAKT
------------------------------
TELEFONY('1234567','3333333')
```


Operacje DML na kolekcji (2)

- Dla tabeli zagnieżdżonej – możliwość operowania na poszczególnych elementach.

```
SELECT w.* FROM projekty, TABLE(wykonawcy) w WHERE nazwa='Z2';
NAZWISKO              PENSJA ETAT           DATA_UR
-------------------- ---------- -------------- --------
Lola                    1000
INSERT INTO TABLE(SELECT wykonawcy FROM projekty WHERE nazwa = 'Z2')
VALUES (Pracownik('Lola'));

DELETE FROM TABLE(SELECT wykonawcy FROM projekty WHERE nazwa = 'Z2') w
WHERE w.nazwisko = 'Tola';

UPDATE TABLE(SELECT wykonawcy FROM projekty WHERE nazwa = 'Z2') w
SET w.pensja = 2500, w.etat = 'KIEROWNIK' WHERE w.nazwisko = 'Lola';

SELECT w.* FROM projekty, TABLE(wykonawcy) w WHERE nazwa='Z2';
NAZWISKO              PENSJA ETAT           DATA_UR
-------------------- ---------- -------------- --------
Lola                    2500 KIEROWNIK
```

Kolekcje w PL/SQL (1)

- Dostępne metody kolekcji:

metoda	opis
EXTEND EXTEND(n) EXTEND(n, i)	Dodaje do kolekcji n pustych elementów (jeden przy braku n), opcjonalnie wypełnia je wartością i-tego elementu.
TRIM TRIM(n)	Usuwa n elementów (jeden przy braku n) od końca kolekcji.
DELETE DELETE(n) DELETE(n, m)	Usuwa wszystkie elementy kolekcji (brak n i m), n-ty element, lub elementy od n-tego do m-tego; ustawienie m i n tylko dla tabeli zagnieżdżonej.
NEXT(n) PRIOR(n)	Zwraca indeks elementu następującego (poprzedzającego) elementu o indeksie n, NULL przy braku elementu.
EXISTS(n)	Testuje istnienie elementu o indeksie n, zwraca NULL przy przekroczeniu zakresu.

Kolekcje w PL/SQL (2)

- Dostępne metody kolekcji (cd):

metoda	opis
FIRST LAST	Zwraca indeks pierwszego (ostatniego) elementu kolekcji.
LIMIT	Zwraca maksymalny zakres kolekcji.
COUNT	Zwraca liczbę elementów kolekcji.

Kolekcje w PL/SQL (3)

- Zwracane wyjątki:
  - COLLECTION_IS_NULL – przy wywołaniu EXISTS dla pustej kolekcji,
  - SUBSCRIPT_BEYOND_COUNT – odwołanie do elementu spoza zakresu w TRIM.

Tabela o zmiennym rozmiarze w PL/SQL

```
DECLARE
  znajomi Telefony := Telefony();
BEGIN
  znajomi.EXTEND;
  znajomi(1) := 'Bolek – 1234567';
  znajomi.EXTEND(znajomi.LIMIT - znajomi.COUNT,1);

  FOR v_i IN 1..znajomi.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(znajomi(v_i));
  END LOOP;

  znajomi.TRIM(4);
  znajomi(2) := 'Lolek – 9876543';
  znajomi(5) := 'Tola – 0102030';

  FOR v_i IN 1..znajomi.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(znajomi(v_i));
  END LOOP;
END;
```


Tabela zagnieżdżona w PL/SQL

```
DECLARE
  bdanych Zespol := Zespol();
  v_j NUMBER;
BEGIN
  bdanych.EXTEND(1);
  bdanych(1) := NEW Pracownik('Bolek');
  bdanych.EXTEND(5,1);
  bdanych.DELETE(2,3);

  FOR v_i IN 1..bdanych.LAST LOOP
    IF bdanych.EXISTS(v_i) THEN
      DBMS_OUTPUT.PUT_LINE(bdanych(v_i).nazwisko);
    END IF;
  END LOOP;

  bdanych(4) := NEW Pracownik('Lolek');
  bdanych(5) := NEW Pracownik('Tola');

  v_j := bdanych.FIRST;
  WHILE (v_j IS NOT NULL) LOOP
    DBMS_OUTPUT.PUT_LINE(bdanych(v_j).nazwisko);
    v_j := bdanych.NEXT(v_j);
  END LOOP;
END;
```

Wynik zapytania jako kolekcja

- Klauzula **BULK COLLECT**:

```
DECLARE
  TYPE tPracownicy IS TABLE OF pracownicy%ROWTYPE;
  prac tPracownicy;
BEGIN
  SELECT * BULK COLLECT INTO prac FROM pracownicy;

  FOR v_i IN 1..prac.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(prac(v_i).nazwisko);
  END LOOP;
END;
```

  - dodatkowa klauzula **SAMPLE(n)** pozwala ograniczyć wynik do n% liczby rekordów w zbiorze.