SGBD - 2e

PL-SQL - Chapitre 5 - Les collections

Daniel Schreurs

6 février 2022

Haute École de Province de Liège

Table des matières du chapitre i

Définir des types *collections*

Table des matières de la section : Définir des types *collections* i

Définir des types *collections* : Définitions

Important

Une *collection* est un ensemble, éventuellement ordonné, d'éléments de même type. Chaque élément est repéré au moyen d'un indice. [1]

1. Dans d'autres langages, on parle aussi de liste ou de vecteur.

Définir des types *collections* : Définitions

Le PL/SQL possède 3 types de collections :

- Les tableaux associatifs (*associative arrays* ou *index-by tables*)
- Les tables imbriquées (*nested tables*)
- Les tableaux prédimensionnés (*variable-size arrays*)

Définir des types *collections* : Les tableaux associatifs

Les tableaux associatifs (*associative arrays* ou *index-by tables*)

Important

Ensemble ordonné d'éléments repérés par un indice de type numérique ou chaîne de caractères. On parle également de table de hachage ou de table PL/SQL.

Définir des types *collections* : Les tables imbriquées

Les tables imbriquées (*nested tables*)

Important

Qui peuvent contenir un ensemble non ordonné d'éléments indicés par des valeurs numériques consécutives.

Définir des types *collections* : Les tableaux prédimensionnés

Les tableaux prédimensionnés (*variable-size arrays*)

Important

Ensemble ordonné d'éléments de même type dont le nombre d'éléments est fixé lors de la déclaration (bien que maintenant, il soit possible de modifier cette limite lors de l'exécution). Les éléments sont indicés par des nombres consécutifs.

Les méthodes associées aux collections

Table des matières de la section : Les méthodes associées aux collections i

Table des matières de la section : Les méthodes associées aux collections ii

Les méthodes associées aux collections : Déclaration

Important

Il n'y a pas de clause d'initialisation ni de constructeur associé à un type tableau associatif !

Syntaxe

```
TYPE type_name
  IS TABLE OF element_type [NOT NULL]
  INDEX BY [PLS_INTEGER |
            BINARY_INTEGER |
            VARCHAR2 (size_limit)];
```

Les méthodes associées aux collections : Déclaration

Exemple 1

```
declare
    TYPE TypeTableMessErreur
        IS TABLE OF VARCHAR2(200)
        INDEX BY BINARY_INTEGER;

    TableMessErreur TypeTableMessErreur;
BEGIN
    -- on peut se servir de TableMessErreur
    null;
END;
```

Les méthodes associées aux collections : Déclaration

Exemple avec ROWTYPE

```
1  create table Emp (nom varchar2(100)); -- Création de la
       table
2
3  declare
4      TYPE TypeLesEmployes
5          IS TABLE OF Emp%ROWTYPE
6          INDEX BY BINARY_INTEGER;
7      TableEmployes TypeLesEmployes;
8  BEGIN
9      -- on peut se servir de TableEmployes
10     null;
11 END;
```

Les méthodes associées aux collections : Initialisation

Exemple avec initialisation

```
declare
    ConstHireDate     CONSTANT NUMBER := -20001;
    ConstNom          CONSTANT NUMBER := -20002;
    ConstHireDateNull CONSTANT NUMBER := -20007;
    TYPE TypeTableMessErreur
        IS TABLE OF VARCHAR2(200)
        INDEX BY BINARY_INTEGER;
    TableMessErreur             TypeTableMessErreur;

BEGIN
    --initialisation
    TableMessErreur(ConstHireDate) :=
            'Date embauche > date du jour';
    TableMessErreur(ConstNom) := 'Nom employé inconnu';
    TableMessErreur(ConstHireDateNull) :=
            'Date embauche inconnue';
END;
```

Les méthodes associées aux collections : Initialisation

Initialiser un tableau associatif

```
1  DECLARE
2      TYPE TypeLesEmployes
3          IS TABLE OF Emp%ROWTYPE
4          INDEX BY BINARY_INTEGER;
5      TableEmployes TypeLesEmployes;
6  BEGIN
7      -- On charge dans le tableau le résultat d'une
           requête !
8      SELECT * BULK COLLECT
9      INTO TableEmployes
10     FROM Emp;
11 EXCEPTION
12     WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
13 END;
```

Les méthodes associées aux collections : Méthode exists

Tester l'existence ou non d'un élément : méthode exists

```
1  DECLARE
2    TYPE TableEmployes IS TABLE OF Employes%ROWTYPE
3      INDEX BY BINARY_INTEGER;
4    LesEmployes TableEmployes;
5  BEGIN
6    SELECT * BULK COLLECT INTO LesEmployes FROM Employes;
7    IF LesEmployes.EXISTS(25) -- Test sur l'indice
8      THEN  DBMS_OUTPUT.PUT_LINE('élément existe');
9      ELSE  DBMS_OUTPUT.PUT_LINE('élément n''existe pas');
10   END IF;
11 EXCEPTION
12   WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
13 END;
```

Les méthodes associées aux collections : Méthode count

Compter le nombre d'éléments d'une collection : méthode count

```
1  DECLARE
2    TYPE TableEmployes IS TABLE OF Employes%ROWTYPE
3      INDEX BY BINARY_INTEGER;
4    LesEmployes TableEmployes;
5  BEGIN
6    DBMS_OUTPUT.PUT_LINE(LesEmployes.COUNT);
7    SELECT * BULK COLLECT INTO LesEmployes FROM Employes;
8    DBMS_OUTPUT.PUT_LINE(LesEmployes.COUNT);
9  EXCEPTION
10   WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
11 END;
```

Les méthodes associées aux collections : Méthode first et last

Déterminer le premier et le dernier indice des éléments d'une collection : méthodes first et last

```
DECLARE
    TYPE TableEmployes IS TABLE OF Employes%ROWTYPE
        INDEX BY BINARY_INTEGER;
    LesEmployes TableEmployes;
BEGIN
    SELECT * BULK COLLECT INTO LesEmployes FROM Employes
        ;
    FOR i IN LesEmployes.FIRST..LesEmployes.LAST
        LOOP
            DBMS_OUTPUT.PUT_LINE(i || ' ' || LesEmployes
                (i).nom);
        END LOOP;
EXCEPTION
    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

Les méthodes associées aux collections : Méthode first et last

Important

Si la collection est vide, `FIRST` et `LAST` donnent `NULL`

Les méthodes associées aux collections : Méthode first et last

L'indice du dernier élément

```
1  DECLARE
2    TYPE TableEmployes IS TABLE OF Employes%ROWTYPE
3      INDEX BY BINARY_INTEGER;
4    LesEmployes TableEmployes;
5    i NUMBER;
6  BEGIN
7    i := LesEmployes.LAST; -- Ici
8    DBMS_OUTPUT.PUT_LINE('i'|| '/' || i || '/');
9  EXCEPTION
10   WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
11 END;
```

Les méthodes associées aux collections : Méthode next

Indices non consécutifs

```
1  -- indices non consécutifs
2  DECLARE
3    TYPE TableEmployes IS TABLE OF Employes.bareme%TYPE
4      INDEX BY VARCHAR2(10);
5    LesEmployes TableEmployes;
6    i VARCHAR2(10);
7  BEGIN
8    i := LesEmployes.FIRST;
9    WHILE i IS NOT NULL LOOP
10     DBMS_OUTPUT.PUT_LINE(i || ' : ' || LesEmployes(i));
11     i := LesEmployes.NEXT(i); -- Ici
12   END LOOP;
13 EXCEPTION
14   WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
15 END;
```

Les méthodes associées aux collections : Méthode delete

- `DELETE` supprime tous les éléments d'une collection
- `DELETE(n)` supprime le nième élément d'une table. Si *n* est `NULL`, `DELETE` n'a pas d'effet
- `DELETE(m, n)` supprime tous les éléments dans la fouchette *m*..*n*. Si m est plus grand que *n*, `DELETE(m, n)` n'a pas d'effet.

Exceptions liées aux collections

Table des matières de la section : Exceptions liées aux collections i

Exceptions liées aux collections : Définitions

- `NO_DATA_FOUND` Un indice désigne un élément supprimé ou un élément qui n'existe pas dans une table PL/SQL
- `VALUE_ERROR` Un indice est null ou ne peut être converti dans le type de l'indice

Exceptions liées aux collections : Définitions

ORA-06502 : PL/SQL : erreur numérique ou erreur sur une valeur

```
1  DECLARE
2    TYPE TableEmployes IS TABLE OF Employes%ROWTYPE
3      INDEX BY BINARY_INTEGER;
4    LesEmployes TableEmployes;
5    i NUMBER;
6  BEGIN
7    FOR i IN LesEmployes.FIRST..LesEmployes.LAST LOOP
8      DBMS_OUTPUT.PUT_LINE(i || '** ' || LesEmployes(i).
           nom);
9    END LOOP;
10 EXCEPTION
11   WHEN VALUE_ERROR THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
12 END;
```

Exceptions liées aux collections : Définitions

Exemple complet : parcourir des résultats de recherches

```
1  DECLARE
2    TYPE TableEmployes IS TABLE OF Employes%ROWTYPE
3      INDEX BY BINARY_INTEGER;
4    LesEmployes TableEmployes;
5    i NUMBER;
6  BEGIN
7    SELECT * BULK COLLECT INTO LesEmployes FROM Employes;
8    FOR i IN LesEmployes.FIRST..LesEmployes.LAST LOOP
9      DBMS_OUTPUT.PUT_LINE(i || '** ' || LesEmployes(i).
           nom);
10   END LOOP;
11 EXCEPTION
12   WHEN VALUE_ERROR THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
13 END;
```

Exceptions liées aux collections : Définitions

Liste complète des exceptions liées aux collections :

EXCEPTION	**Déclenchée lorsque...**
`COLLECTION_IS_NULL`	On utilise une collection atomicaly null
`NO_DATA_FOUND`	Un indice désigne un élément supprimé ou un élément qui n'existe pas dans une table PL/SQL
`SUBSCRIPT_BEYOND_COUNT`	Un indice dépasse le nombre d'éléments de la collection
`SUBSCRIPT_OUTSIDE_LIMIT`	Un indice est en dehors de la fourchette permise
`VALUE_ERROR`	Un indice est null ou ne peut être converti dans le type de l'indice