Tipos de Dados Compostos

Programação em Banco de Dados

Outline

- Registro PL/SQL (PL/SQL Record) -- já visto
- Tipo de Objeto (Object Type)
- Coleção (VARRAY, Nested Table e PL/SQL Table)
- System Reference Cursor

Tipo de Objeto (Object Type)

- É um tipo abstrato de dados (TAD)
- TAD é um tipo de dado definido pelo usuário que encapsula propriedades (atributos) e comportamento (métodos)
- Corresponde SOMENTE ao “molde” de um objeto
- Não pode armazenar dados
- Não aloca espaço de armazenamento
- Pode ser herdado por outro Object Type, desde que seja NOT FINAL (default é FINAL)

Tipo de Objeto (Object Type)

- Um Object Type é um objeto definido ao nível de esquema com 3 componentes:
  - Nome → obrigatório
  - Atributos → obrigatório
  - Métodos → facultativo (vem com método construtor)

Tipo de Objeto (Object Type)

- Contém uma coleção de campos de elementos como se fosse uma linha de tabela.

```
CREATE OR REPLACE TYPE president_object IS OBJECT
(salutation VARCHAR(20),
 name VARCHAR2(10));
```

Object Type Body

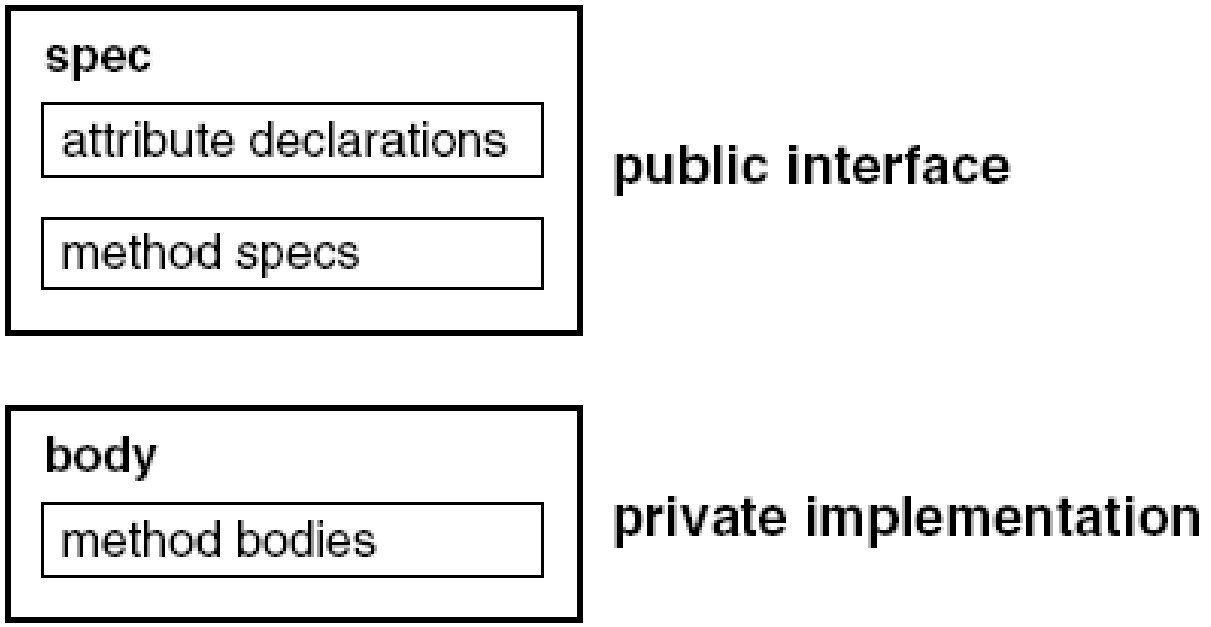
- O código que especifica um Object Type normalmente vem acompanhado de outro que **implementa** o corpo do objeto.
  - Corpo do tipo de objeto (Object Type Body).

- O nome do Object Type Body é o mesmo do Object Type.

- É possível existir um Object Type **sem** Object Body, embora o mais comum é existirem ambos.

Object Type e Object Type Body

- Estrutura:



Object Type e Object Type Body

```
CREATE OR REPLACE TYPE
president_object IS OBJECT
(salutation VARCHAR(20),
 name VARCHAR2(10),
MEMBER FUNCTION full RETURN
varchar2);
```

Todo Object Type possui um método construtor com o mesmo nome do Object Type.

```
CREATE OR REPLACE TYPE BODY
president_object IS
  MEMBER FUNCTION full
RETURN VARCHAR2 IS
  BEGIN
    return self.salutation || ' ' ||
self.name;
  END;
END;
```

Como usar um Tipo de Objeto (Object Type)?

1. Tipo de dados de uma coluna de uma tabela relacional.
2. Estrutura de uma tabela (Object Table). -- repositório de objetos
3. Tipo de dados de um parâmetro de uma função ou procedure.
4. Tipo de dados de uma variável em um bloco.
5. Tipo de dados de retorno de uma função.

1) Object Type como Tipo de Dados de uma Coluna

```
CREATE TABLE my_tab65
(id NUMBER,
full president_object,
dob DATE);

INSERT INTO my_tab65 VALUES(10,president_object('Dr.','No'),SYSDATE);
```

2) Object Type como Estrutura de uma Tabela (Object Table)

CREATE TABLE president_tab OF **president_object**;

INSERT INTO president_tab VALUES (president_object('Dr.','No'));

3) Object Type como Tipo de Dados de um Parâmetro de uma Função ou Procedure

```
CREATE OR REPLACE PROCEDURE my_proc98 (p_president IN
president_object) IS
BEGIN
  dbms_output.put_line(p_president.salutation || ' ' || p_president.name);
END;
```

4) Object Type como Tipo de Dados de uma Variável em um Bloco

```
DECLARE
  president PRESIDENT_OBJECT := president_object('Mr.','Lincoln');
BEGIN
  dbms_output.put_line(president.salutation||' '||president.name);
END;
```

Mr. Lincoln

5) Object Type como Tipo de Dados de Retorno de uma Função

```
CREATE OR REPLACE FUNCTION my_func35 (p_salutation VARCHAR2,
p_name VARCHAR2) RETURN president_object
IS
  president PRESIDENT_OBJECT;
BEGIN
  president := president_object(p_salutation,p_name);
  RETURN(president);
END;
```

Object Type vs. Registros

- Um tipo de dados Registro (IS RECORD) geralmente contém uma coleção de campos relacionados, similar a estrutura de uma linha de uma tabela.

- Uma coleção de registros é chamada de Tabela PL/SQL (em memória).
  - A ser visto adiante.

Object Type vs. Registros

- Uma variável Object Type é similar a uma variável RECORD, caso o Object Type tenha sido criado **sem métodos**.
  - A atribuição de valores a uma variável Object Type deve usar o método (função) construtor.
  - A ordem dos valores deve obedecer à ordem dos atributos do Object Type.

Object Type vs. Registros

- Object Types podem ser usados para definir o tipo de dados de uma coluna de uma tabela (Object Column) e estruturas de tabela (Object Table); registros não.

- Object Types podem ser usados como tipo de dados de parâmetros em funções ou procedures chamadas a partir de SQL e PL/SQL; registros só funcionam em PL/SQL.

Object Type vs. Registros

- Coleções de Object Types podem definir tipos de dados de colunas (Nested Table) mas não estruturas de tabela.

- Coleções de Object Types podem ser usadas em SQL e PL/SQL; coleções de registros só em PL/SQL.

Introdução

- Registro PL/SQL (PL/SQL Record) -- já visto
- Tipo de Objeto (Object Type)
- Coleção (VARRAY, Nested Table e PL/SQL Table)
- System Reference Cursor

Coleção

- É uma estrutura de dados que armazena um conjunto de dados relacionados.

- O Oracle suporta três tipos de coleção:
  - Dois de tipos de dados SQL e PL/SQL:
    - **Varray** (Array SQL) e **Nested table**: ambos são tipos de objeto definidos ao nível de esquema.
  - O terceiro de tipo de dado apenas PL/SQL:
    - **Associative array** (PL/SQL table ou index-by table): é implementado como uma lista e pode ser indexada com números ou strings.

Coleção

- O SGBD Oracle implementa **arrays** com o tipo de dados **Varray** e **listas** com o tipo de dados **Nested Table** (tabela aninhada).

- Ambas as estruturas podem ser de tipo de dado escalar ou composto.

- Uma coleção de tipo de dado **escalar** é um *Attribute Data Type* (**ADT**).

- Uma coleção de tipo de dado **composto** é um *User-Defined Type* (**UDT**).

Coleções SQL vs. Tabelas PL/SQL

	SQL Collections	Associative Arrays
Scope	May be defined in SQL or PL/SQL scope.	Are defined only in PL/SQL scope.
Initialization	Require initialization before their first use.	Don't require initialization.
Assignment	Preallocate space before assigning values. You can preallocate space and assign values for more than one element at a time.	Don't need to allocate space because you manually assign indexes and values one row at a time.
Index	Use a sequential set of integers as index values (at least initially for the table data types), which makes SQL collections densely populated arrays and lists.	Use an integer or string as the index, and the index value may be in any order you like, which makes associative arrays sparsely populated lists.
Base Data Type	Use any SQL scalar data type or UDT object type.	Use any SQL or PL/SQL scalar data type or a PL/SQL record type.

Mapeamentos para Tipos Compostos PL/SQL

Non PL/SQL Composite Type	Equivalent PL/SQL Composite Type
Hash table	Associative array
Unordered table	Associative array
Set	Nested table
Bag	Nested table
Array	Varray

Coleção VARRAY

- É uma estrutura de dados uni-dimensional que possui um número máximo de elementos (pré-definido).
- Os elementos são de um mesmo tipo de dados.
- Pode armazenar dados de tipos de dados escalares ou compostos.
- Sintaxe:

  TYPE type_name IS VARRAY (size_limit) OF data_type [NOT NULL];

Coleção VARRAY – Escopo: PL/SQL

```
CREATE OR REPLACE TYPE sql_varray IS VARRAY(3) OF VARCHAR2(20);

DECLARE
  lv_stooges SQL_VARRAY := sql_varray('Moe','Larry');
BEGIN
  dbms_output.put_line('Count: '||lv_stooges.COUNT);
  dbms_output.put_line('Limit: '||lv_stooges.LIMIT);
   lv_stooges.EXTEND;
  dbms_output.put_line('Count: '||lv_stooges.COUNT);
  dbms_output.put_line('Limit: '||lv_stooges.LIMIT);
  lv_stooges(lv_stooges.COUNT) := 'Curly';
  FOR i IN 1..lv_stooges.COUNT LOOP
    dbms_output.put_line(lv_stooges(i));
  END LOOP;
END;
```

- Count: 2
- Limit: 3
- Count: 3
- Limit: 3
- Moe
- Larry
- Curly

Coleção VARRAY – Escopo: SQL

```
CREATE OR REPLACE TYPE sql_varray IS
VARRAY(3) OF VARCHAR2(20);

CREATE TABLE tab16 (id NUMBER,
elementos SQL_VARRAY);

INSERT INTO tab16 VALUES (10,
sql_varray('Moe','Larry','Curly'));
```

```
SELECT t1.id, t2.column_value AS "Stooges"
FROM tab16 t1, TABLE(t1.elementos) t2
ORDER BY 2;
```

ID	Stooges
10	Curly
10	Larry
10	Moe

Coleção Tabela Aninhada (Nested Table)

- É uma estrutura de dados uni-dimensional **sem limite** em relação ao número de elementos.
- Os elementos são do mesmo tipo de dados.
- Pode armazenar dados de tipos de dados escalares ou compostos.
- Sintaxe:

TYPE type_name IS TABLE OF data_type [NOT NULL];

Coleção Tabela Aninhada (Nested Table) – Escopo: PL/SQL

```
CREATE OR REPLACE TYPE sql_table IS TABLE OF VARCHAR2(20);

CREATE OR REPLACE FUNCTION add_element
(pv_element VARCHAR2) RETURN SQL_TABLE
IS
  lv_table SQL_TABLE := sql_table();
BEGIN
  lv_table.EXTEND;
  lv_table(lv_table.COUNT) := pv_element;
  RETURN lv_table;
END;
```

Coleção Tabela Aninhada (Nested Table) – Escopo: PL/SQL

```
DECLARE
  TYPE plsql_table IS TABLE OF VARCHAR2(20);
  lv_table PLSQL_TABLE := plsql_table('Aragorn','Faramir','Boromir');
BEGIN
  FOR i IN lv_table.FIRST..lv_table.LAST LOOP
    dbms_output.put_line(lv_table(i));
  END LOOP;
END;
```

Coleção Tabela Aninhada (Nested Table) – Escopo: SQL

```
CREATE OR REPLACE TYPE sql_table IS TABLE OF VARCHAR2(20);

CREATE TABLE tab78 (id NUMBER, elementos SQL_TABLE)
NESTED TABLE elementos STORE AS elementos_nt;

INSERT INTO tab78 VALUES (1, sql_table('Aragorn','Faramir','Boromir'));
```

```
SELECT t1.id, t2.column_value AS "Dúnedain"
FROM tab78 t1, TABLE(t1.elementos) t2
ORDER BY 1;
```

ID	Dúnedain
1	Aragorn
1	Boromir
1	Faramir

Arrays Associativos (Tabelas PL/SQL)

- São estruturas de dados uni-dimensionais compostas e suportam os mesmos tipos de dados das Coleções SQL (escalar ou compostos).
- São também conhecidos como PL/SQL tables (Tabelas PL/SQL).
- Não é possível definir uma coleção multidimensional.
- **Não podem ser utilizados como tipo de dados em colunas.**
  - **Apenas como estruturas de programação PL/SQL.**

Arrays Associativos (Tabelas PL/SQL)

- Não requerem inicialização e não possuem construtor.
- Não é necessário alocar espaço antes de inserir um valor.
- Podem ser indexadas numericamente (inteiro negativo, positivo ou zero) ou com strings únicos de tamanho variável.
- %ROWTYPE, record type e valores de retorno de Object Type são convertidos implicitamente para estruturas de arrays associativos.

Arrays Associativos (Tabelas PL/SQL)

- Sintaxe

CREATE OR REPLACE TYPE type_name AS TABLE OF base_type [NOT NULL] ***INDEX BY*** *[PLS_INTEGER | BINARY_INTEGER | VARCHAR2(size)];*

Arrays Associativos (Tabelas PL/SQL)

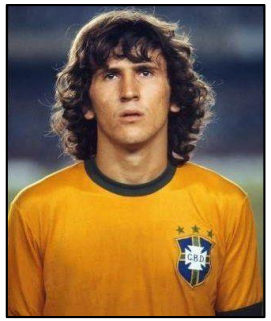
```
DECLARE
  TYPE suit_table IS TABLE OF VARCHAR2(7 CHAR) INDEX BY BINARY_INTEGER;
  lv_suit suit_table;
BEGIN
  lv_suit(1) := 'Club';
  lv_suit(2) := 'Heart';
  lv_suit(3) := 'Diamond';
  lv_suit(4) := 'Spade';
  FOR i IN lv_suit.FIRST..lv_suit.LAST LOOP
    dbms_output.put_line(lv_suit(i));
  END LOOP;
END;
```

Saída:

Club
Heart
Diamond
Spade

Arrays Associativos (Tabelas PL/SQL)

```
CREATE OR REPLACE
TYPE player_object IS OBJECT
(nick VARCHAR2(8),
name VARCHAR(30));
```



```
DECLARE
  TYPE player_table IS TABLE OF player_object INDEX BY
BINARY_INTEGER;
  lv_player PLAYER_TABLE;
BEGIN
  lv_player(1) := player_object('Fenomeno','Ronaldo Nazario');
  lv_player(2) := player_object('Zico','Arthur Antunes');
  lv_player(3) := player_object('Bizu','Claudio Tavares');
  FOR i IN 1..lv_player.COUNT LOOP
    dbms_output.put_line(lv_player(i).nick);
  END LOOP;
END;
/
```

Saída:

Fenomeno
Zico
Bizu

Arrays Associativos (Tabelas PL/SQL)

```
DECLARE
  current VARCHAR2(5);
  TYPE card_table IS TABLE OF NUMBER INDEX BY VARCHAR2(5);
  lv_card CARD_TABLE;
BEGIN
  lv_card('One') := 1;
  lv_card('Two') := 2;
  lv_card('Queen') := 12;
  lv_card('King') := 13;
  current := lv_card.FIRST; -- First alphabetical key.
  WHILE (current <= lv_card.LAST) LOOP -- The WHILE loop reads through the indexes from lowest to highest, which means alphabetically.
    dbms_output.put_line('Values ['||current||']['||lv_card(current)||']');
    current := lv_card.NEXT(current);
  END LOOP;
END;
```

Saída:

Values [King][13]
Values [One][1]
Values [Queen][12]
Values [Two][2]

Introdução

- Registro PL/SQL (PL/SQL Record) -- já visto
- Tipo de Objeto (Object Type)
- Coleção (VARRAY, Nested Table e PL/SQL Table)
- System Reference Cursor

System Reference Cursor

- É um ponteiro para um *result set* contido em uma região de memória (aka *context area*) localizada na *Program Global Area* (PGA).

- Uma *context area* mantém informações de uma consulta:
  - Linhas retornadas pela consulta.
  - Quantidade de linhas processadas pela consulta.
  - Um apontador para a consulta analisada na *query work area*.

- É usado quando se deseja consultar dados a partir de um programa e processar de outro, especialmente quando os programas estão escritos em LPs diferentes.

System Reference Cursor

- Um System reference cursor é um tipo de dados unicamente PL/SQL.
- Pode ser de dois tipos:
  - *Cursor fracamente tipado*: herda o tipo em tempo de execução.
  - *Cursor fortemente tipado*: é especificado em tempo de compilação.
- Pode ser definido em blocos anônimos e nomeados.
- É mais utilizado em especificações de pacotes pois pode ser compartilhado pelos programas.

System Reference Cursor

```
VARIABLE sv_refcursor REFCURSOR
DECLARE
  TYPE weakly_typed IS REF CURSOR; -- Declare a weakly typed reference cursor.
  lv_refcursor WEAKLY_TYPED; -- Declare a local variable of the weakly typed reference cursor.
BEGIN
  OPEN lv_refcursor FOR -- Open the reference cursor.
  SELECT job_id, COUNT(*)
  FROM employees
  GROUP BY job_id;
  -- Assign the reference cursor to a SQL*Plus session variable.
  :sv_refcursor := lv_refcursor;
END;
```

```
SELECT :sv_refcursor
FROM dual;


:SV_REFCURSOR
--------------------
CURSOR STATEMENT : 1
CURSOR STATEMENT : 1


JOB_ID      COUNT(*)
---------- ----------
AC_ACCOUNT        1
AC_MGR            1
AD_PRES           2
…
```