**\*OBJETO:**

CREATE OR REPLACE TYPE nome\_tipo AS OBJECT(

[lista de atributos]

[lista de métodos]

)[NOT INSTANTIABLE];

**EX.:** CREATE OR REPLACE TYPE tp\_pessoa AS OBJECT (

id NUMBER,

pri\_nome VARCHAR2(20),

ult\_nome VARCHAR2(25),

email VARCHAR2(25),

fone VARCHAR2(12),

MEMBER PROCEDURE exibir\_detalhes ( SELF IN OUT tp\_pessoa),

MAP MEMBER FUNCTION get\_id RETURN NUMBER

);

OBS.: Métodos membro: têm acesso aos dados da instância do objeto

Order: realiza comparações object-to-object (não pode ter se houver um Member)

Member: fornece a base para comparar objetos, mapeando as instâncias dos objetos em um dos tipos escalares DATE, NUMBER, VARCHAR2 (não pode ter se houver um Order)

**\*MÉTODO:**

CREATE [OR REPLACE] TYPE BODY nome\_tipo AS

[lista de subprogramas (procedimento, função ou construtor)]

[lista de funções MAP ou ORDER]

END ;

**EX.1:** CREATE OR REPLACE TYPE BODY tp\_pessoa AS

MEMBER PROCEDURE exibir\_detalhes ( SELF IN OUT tp\_pessoa) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(id) || ' ' || pri\_nome || ' ' || ult\_nome);

DBMS\_OUTPUT.PUT\_LINE(email || ' ' || fone);

END;

MAP MEMBER FUNCTION get\_id RETURN NUMBER IS

BEGIN

RETURN id;

END;

END;

**EX.2:** CREATE OR REPLACE TYPE BODY tp\_retangulo AS

CONSTRUCTOR FUNCTION tp\_retangulo (x NUMBER, y NUMBER, Largura NUMBER altura NUMBER) RETURN SELF AS RESULT IS

BEGIN

SELF.setXY(x, y); setLargura(largura); setAltura(altura); RETURN;

END;

MEMBER FUNCTION getLargura RETURN NUMBER IS

BEGIN

RETURN largura;

END;

MEMBER FUNCTION getAltura RETURN NUMBER IS

*semelhante a getLargura()*

MEMBER PROCEDURE setLargura (newLargura NUMBER) IS

BEGIN

largura := newLargura;

END;

MEMBER PROCEDURE setAltura(newAltura NUMBER) IS

*semelhante a setLargura()*

OVERRIDING MEMBER PROCEDURE desenhar IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Desenhar retângulo:(' || SELF.getX() ||',' || SELF.getY() || '), largura ' || getLargura() || ', altura ' || getAltura());

END;

END;

**\*ALTERAR TIPO:** CASCADE propaga as mudanças para todos os dependents

ALTER TYPE tp\_pessoa ADD ATTRIBUTE ( sexo VARCHAR2(1) ) CASCADE ;

ALTER TYPE tp\_pessoa DROP MAP MEMBER FUNCTION get\_id RETURN NUMBER INVALIDATE; (INVALIDATE invalida todos os tipos dependentes)

ALTER TYPE tp\_pessoa NOT FINAL CASCADE; (permite especializar o tipo tp\_pessoa)

ALTER TYPE tp\_pessoa MODIFY ATTRIBUTE sexo VARCHAR2(10) CASCADE ; (modifica tamanho do atributo)

**\*TIPO:**

CREATE TYPE nome\_tipo [UNDER outro\_tipo](

[lista de atributos]

)[NOT FINAL];

OBS.: NOT FINAL indica que outro tipo pode herdar dele

UNDER é usado para indicar que herda de outro tipo

ALTER TYPE tp\_ponto **NOT FINAL CASCADE**;

**EX.:** CREATE TYPE tp\_funcionario **UNDER** tp\_pessoa(

depto\_id NUMBER,

funcao VARCHAR2(30),

salario NUMBER

) NOT FINAL;

**\*REFERÊNCIA:**

OBS.: IS DABGLING é quando o objeto indicado pelo REF foi removido (é nulo)

WITH ROWID REFERENCES serve para garantir integridade

**EX.1:** CREATE OR REPLACE TYPE tp\_cliente as OBJECT(

cod\_cli VARCHAR (3),

nm\_cli VARCHAR (60));

CREATE TABLE tb\_cliente OF tp\_cliente(

cod\_cli PRIMARY KEY,

nm\_cli NOT NULL);

CREATE OR REPLACE TYPE tp\_dependente as OBJECT(

cod\_dep VARCHAR(3),

nm\_dep VARCHAR (60),

ref\_titular **REF** tp\_cliente);

CREATE TABLE tb\_dependente OF tp\_dependente(

cod\_dep PRIMARY KEY,

nm\_dep NOT NULL,

ref\_titular **SCOPE IS** tb\_cliente);

INSERT INTO tb\_cliente VALUES (‘C1’, ‘Rita’);

INSERT INTO tb\_cliente VALUES (‘C2’, ‘Ana’);

INSERT INTO tb\_dependente

SELECT ‘D1’, ‘Paulo’, REF (C) FROM tb\_cliente C WHERE cod\_cli = ‘C1’;

INSERT INTO tb\_dependente (SELECT ‘D2’, ‘Pedro’, REF (C) FROM tb\_cliente C

WHERE cod\_cli = ‘C2’);

SELECT REF(D) FROM tb\_dependente D WHERE D. nm\_dep = ‘Paulo’;

SELECT D.ref\_titular.cod\_cli AS cod\_cliente, D.ref\_titular.nm\_cli AS nm\_cliente,

D.nm\_dep AS nm\_dependente FROM tb\_dependente D;

**EX.2:** DROP TABLE tb\_dependente;

CREATE TABLE tb\_dependente OF tp\_dependente(

cod\_dep PRIMARY KEY,

nm\_dep NOT NULL,

ref\_titular **WITH ROWID REFERENCES** tb\_cliente);

**\*DESREFERENCIAR:**

**EX.:** SELECT DEREF(D.ref\_titular) deref\_titular, D.nm\_dep nm\_dependente

FROM tb\_dependente D;

**\*VARRAY:** tamanho fixo

**EX.1:** CREATE OR REPLACE TYPE tp\_fone AS OBJECT (

cod\_pais VARCHAR2(2),

cod\_area VARCHAR2(2),

numero VARCHAR2(8));

CREATE OR REPLACE TYPE tp\_va\_fone AS **VARRAY**(5) OF tp\_fone;

CREATE TABLE tb\_lista\_fone\_departamento(

cod\_depto NUMBER(5),

lista\_fone tp\_va\_fone);

INSERT INTO tb\_lista\_fone\_departamento VALUES (100,

tp\_va\_fone (tp\_fone(‘55', ‘81', ‘22222222'),

tp\_fone(‘55', ‘81', ‘33333333'),

tp\_fone(‘55', ‘81', ‘44444444')));

**\*NESTED TABLE:** tamanho variável

**EX.1:** CREATE TYPE tp\_nt\_fone AS **TABLE** OF tp\_fone;

CREATE TABLE tb\_lista\_fone\_contato(

cod\_contato NUMBER(5),

lista\_fone tp\_nt\_fone)

NESTED TABLE lista\_fone STORE AS tb\_lista\_fone;

INSERT INTO tb\_lista\_fone\_contato VALUES (001,

tp\_nt\_fone (tp\_fone(‘55', ‘81', ‘55555555'),

tp\_fone(‘55', ‘81', ‘66666666')));

**EX.2:** CREATE OR REPLACE TYPE tp\_nt\_fone AS **TABLE** OF tp\_fone;

CREATE OR REPLACE TYPE tp\_lista\_fone\_contato AS OBJECT(

cod\_contato NUMBER(5),

lista\_fone tp\_nt\_fone);

CREATE TABLE tb\_lista\_fone\_contato OF tp\_lista\_fone\_contato;

NESTED TABLE lista\_fone STORE AS tb\_lista\_fone;

INSERT INTO tb\_lista\_fone\_contato VALUES (001,

tp\_nt\_fone (tp\_fone(‘55', ‘81', ‘55555555'),

tp\_fone(‘55', ‘81', ‘66666666')));

SELECT \*FROM **TABLE**(SELECT C.lista\_fone FROM tb\_lista\_fone\_contato C

WHERE C.cod\_contato = 001);

SELECT C.cod\_depto, T.\* FROM tb\_lista\_fone\_departamento C, **TABLE**(C.lista\_fone) T;

**EX.3:** CREATE OR REPLACE TYPE tp\_conta; *-- Tipo incomplete*

CREATE OR REPLACE TYPE tp\_nt\_ref\_conta AS TABLE OF **REF** tp\_conta;

CREATE OR REPLACE TYPE tp\_cliente AS OBJECT (

cod\_cli NUMBER(4),

nm\_cli VARCHAR2(60),

lista\_conta tp\_nt\_ref\_conta);

CREATE OR REPLACE TYPE tp\_nt\_ref\_cliente AS TABLE OF **REF** tp\_cliente;

CREATE OR REPLACE TYPE tp\_conta AS OBJECT(

num\_cc NUMBER(4),

saldo NUMBER(8,2),

lista\_cliente tp\_nt\_ref\_cliente);

CREATE TABLE tb\_cliente OF tp\_cliente (cod\_cli PRIMARY KEY) NESTED TABLE lista\_conta

STORE AS tb\_lista\_conta;

CREATE TABLE tb\_conta OF tp\_conta (num\_cc PRIMARY KEY) NESTED TABLE lista\_cliente

STORE AS tb\_lista\_cliente;

INSERT INTO tb\_cliente VALUES ( 0001, 'Jorge', tp\_nt\_ref\_conta() );

INSERT INTO tb\_cliente VALUES ( 0002, 'Rosa', tp\_nt\_ref\_conta() );

INSERT INTO tb\_conta VALUES ( 1111, 1000.00,

tp\_nt\_ref\_cliente( (SELECT REF(C) FROM tb\_cliente C WHERE C.cod\_cli = 0001),

(SELECT REF(C) FROM tb\_cliente C WHERE C.cod\_cli = 0002)));