BD07.

UT07 Uso de bases de datos objeto-relacionales.

Profesor: Valentín Rebollada Casado.

Alumna: Cherry Reynoso Catalán

DAM Distancia. Base de datos

28/04/2024

Cherry Reynoso Catalán

ÍNDICE

ENUNCIADO TAREA 01..... 2
1. CREA EL TIPO DE OBJETOS "Personal" CON LOS SIGUIENTES ATRIBUTOS:..... 2
CREA, COMO TIPO HEREDADO DE "Personal", EL TIPO DE OBJETO "Responsable" CON LOS SIGUIENTES ATRIBUTOS:..... 2
CREA EL TIPO DE OBJETO "Zonas" CON LOS SIGUIENTES ATRIBUTOS:..... 3
CREA, COMO TIPO HEREDADO DE "Personal", EL TIPO DE OBJETO "Comercial" CON LOS SIGUIENTES ATRIBUTOS:..... 3

2. CREA UN MÉTODO CONSTRUCTOR PARA EL TIPO DE OBJETOS "Responsable", EN EL QUE SE INDIQUEN COMO PARÁMETROS EL código, nombre, primer apellido, segundo apellido y tipo. ESTE MÉTODO DEBE ASIGNAR AL ATRIBUTO apellidos LOS DATOS DE PRIMER APELLIDO Y SEGUNDO APELLIDO QUE SE HAN PASADO COMO PARÁMETROS, UNIÉNDOLOS CON UN ESPACIO ENTRE ELLOS..... 4

3. CREA UN MÉTODO getNombreCompleto PARA EL TIPO DE OBJETOS Responsable QUE PERMITA OBTENER SU NOMBRE COMPLETO CON EL FORMATO apellidos nombre..... 5

4. CREA UN TABLA TablaResponsables DE OBJETOS Responsable. INSERTA EN DICHA TABLA DOS OBJETOS Responsable..... 6
EL SEGUNDO OBJETO "Responsable" DEBES CREARLO USANDO EL MÉTODO CONSTRUCTOR QUE HAS REALIZADO ANTERIORMENTE. DEBES USAR LOS SIGUIENTES DATOS:..... 7

7. OBTENER, DE LA TABLA TablaComerciales, EL Comercial QUE TIENE EL CÓDIGO 100, ASIGNÁNDOSELO A UNA VARIABLE unComercial..... 10

8. MODIFICA EL CÓDIGO DEL COMERCIAL GUARDADO EN ESA VARIABLE unComercial ASIGNANDO EL VALOR 101, Y SU zona DEBE SER LA SEGUNDA QUE SE HABÍA CREADO ANTERIORMENTE. INSERTA ESE COMERCIAL EN LA TABLA TablaComerciales..... 10

9. CREA UN MÉTODO MAP ordenarZonas PARA EL TIPO Zonas. ESTE MÉTODO DEBE RETORNAR EL NOMBRE COMPLETO DEL Responsable AL QUE HACE REFERENCIA CADA zona. PARA OBTENER EL NOMBRE DEBES UTILIZAR EL MÉTODO getNombreCompleto QUE SE HA CREADO ANTERIORMENTE..... 11

10. REALIZA UNA CONSULTA DE LA TABLA TablaComerciales ORDENADA POR zonaComercial PARA COMPROBAR EL FUNCIONAMIENTO DEL MÉTODO MAP..... 12

SCRIPT COMPLETO..... 13

Cherry Reynoso Catalán

ENUNCIADO TAREA 01.

Realiza los siguientes ejercicios. Antes de comenzar lee con detalle el enunciado de la tarea íntegramente y consulta con tu tutor o tutora las dudas que te puedan surgir.

Los ejercicios se basan en un Sistema de Información de una empresa de ventas que gestiona los comerciales y las zonas que tienen asignadas. Una zona puede estar asignada a varios comerciales. Cada zona tiene un único jefe que es el responsable de zona, pudiendo encargarse de varias zonas.

1. CREA EL TIPO DE OBJETOS "Personal" CON LOS SIGUIENTES ATRIBUTOS:

```
codigo INTEGER,
dni VARCHAR2(10),
nombre VARCHAR2(30),
apellidos VARCHAR2(30),
sexo VARCHAR2(1),
fecha_nac DATE
```

```
/*1.CREA EL TIPO DE OBJETOS "Personal"*/
CREATE OR REPLACE TYPE Persona1 AS OBJECT (
        -- Aquí indicaremos los atributos de nuestro TIPO-OBJETO
"Persona1"
        codigo INTEGER,
        dni VARCHAR(10),
        nombre VARCHAR(30),
        apellidos VARCHAR(20),
        sexo VARCHAR(1),
        fecha_nac DATE

        -- Si hubiese métodos, irían aquí.
    ) NOT FINAL; -- Para que otro tipo objeto pueda heredar de
"Persona1" debe tener la propiedad NOT FINAL
    /
```

CREA, COMO TIPO HEREDADO DE "Personal", EL TIPO DE OBJETO "Responsable" CON LOS SIGUIENTES ATRIBUTOS:

```
tipo  CHAR ,
antiguedad INTEGER
```

```
   --Crea como tipo heredado de "Persona 1", el tipo-objeto
"Responsable".
```

```
    --Usamos la palabra reservada UNDER para indicar que hereda
    CREATE OR REPLACE TYPE Responsable UNDER Persona1 (
        tipo CHAR,
        antiguedad INTEGER
    );
    /
```

CREA EL TIPO DE OBJETO "Zonas" CON LOS SIGUIENTES ATRIBUTOS:

codigo INTEGER,
nombre VARCHAR2(20),
refRespon REF Responsable,
codigoPostal CHAR(5),

```
    -- Crea tipo de objeto "Zonas" con referencia a objeto
"Responsable"
    CREATE OR REPLACE TYPE Zonas AS OBJECT(
        codigo INTEGER,
        nombre VARCHAR(20),
        refRespon REF Responsable, -- Referimos a objeto "Responsable"
        codigoPostal CHAR(5)
    );
    /
```

CREA, COMO TIPO HEREDADO DE "Personal", EL TIPO DE OBJETO "Comercial" CON LOS SIGUIENTES ATRIBUTOS:

zonaComercial Zonas

```
    -- Crea como tipo heredado de "Persona1", el tipo de objeto
"Comercial"
    CREATE OR REPLACE TYPE Comercial UNDER Persona1(
        ZonaComercial Zonas --ZonaComercial es un atributo de tipo
"Zonas" (como una instancia)
    );
    /
```

2. CREA UN MÉTODO CONSTRUCTOR PARA EL TIPO DE OBJETOS "`Responsable`", EN EL QUE SE INDIQUEN COMO PARÁMETROS EL `código, nombre, primer apellido, segundo apellido y tipo`. ESTE MÉTODO DEBE ASIGNAR AL ATRIBUTO `apellidos` LOS DATOS DE PRIMER APELLIDO Y SEGUNDO APELLIDO QUE SE HAN PASADO COMO PARÁMETROS, UNIÉNDOLOS CON UN ESPACIO ENTRE ELLOS.

```
CREATE OR REPLACE TYPE Responsable UNDER Persona1 (
        tipo CHAR,
        antiguedad INTEGER,
        /* 2.CREA UN MÉTODO CONSTRUCTOR PARA EL TIPO DE OBJETOS
"Responsable",
        EN EL QUE SE INDIQUEN COMO PARÁMETROS EL código, nombre, primer
apellido, segundo apellido y tipo.*/
        -- Especificación de atributos del METODO CONSTRUCTOR. Usamos
la palabra CONSTRUCTOR para indicar que es un método constructor.
        CONSTRUCTOR FUNCTION Responsable (
        codigo INTEGER,
        nombre VARCHAR2, -- Usamos VARCHAR2 con el método constructor
        primerApellido VARCHAR2,
        segundoApellido VARCHAR2,
        tipo CHAR
        )
        RETURN SELF AS RESULT
    );
    /

-- Creamos el cuerpo del tipo Objeto Responsable.
CREATE OR REPLACE TYPE BODY Responsable AS
    CONSTRUCTOR FUNCTION Responsable (
                codigo INTEGER,
                nombre VARCHAR2, -- Usamos VARCHAR2 con el método
constructor.
                primerApellido VARCHAR2,
                segundoApellido VARCHAR2,
                tipo CHAR)
            RETURN SELF AS RESULT IS
        BEGIN
        /*2.1 ESTE MÉTODO DEBE ASIGNAR AL ATRIBUTO apellidos LOS DATOS
DE PRIMER APELLIDO Y SEGUNDO APELLIDO QUE SE HAN PASADO COMO
PARÁMETROS,
            UNIÉNDOLOS CON UN ESPACIO ENTRE ELLOS*/
```

```
            -- Ahora asignamos los valores correspondientes a los
atributos usando SELF
            SELF.codigo := codigo;
            SELF.nombre := nombre;
            SELF.apellidos:= primerApellido || ' ' || segundoApellido;
-- Aplicamos la concatenación en el SELF.apellidos.
            SELF.tipo := tipo;
            RETURN;
        END; -- Finalizamos el bloque del MÉTODO CONSTRUCTOR
```

3. CREA UN MÉTODO `getNombreCompleto` PARA EL TIPO DE OBJETOS `Responsable` QUE PERMITA OBTENER SU NOMBRE COMPLETO CON EL FORMATO `apellidos nombre`

```
CREATE OR REPLACE TYPE Responsable UNDER Persona1 (
        tipo CHAR,
        antiguedad INTEGER,
        CONSTRUCTOR FUNCTION Responsable (
        codigo INTEGER,
        nombre VARCHAR2, -- Usamos VARCHAR2 con el método constructor
        primerApellido VARCHAR2,
        segundoApellido VARCHAR2,
        tipo CHAR
        )
        RETURN SELF AS RESULT,
        /*3. CREA UN MÉTODO getNombreCompleto PARA EL TIPO DE OBJETOS
Responsable
        QUE PERMITA OBTENER SU NOMBRE COMPLETO CON EL FORMATO apellidos
nombre*/

        -- Especificación del método con MEMBER FUNCTION para que
retorne un valor (MEMBER PROCEDURE no lo hace)
        MEMBER FUNCTION getNombreCompleto RETURN VARCHAR2 -- Usamos
"RETURN" para que nos devuelva el valor.
    );
    /

-- Creamos el cuerpo del tipo Objeto Responsable.
CREATE OR REPLACE TYPE BODY Responsable AS
    CONSTRUCTOR FUNCTION Responsable (
                codigo INTEGER,
```

```
                nombre VARCHAR2, -- Usamos VARCHAR2 con el método
constructor.
                primerApellido VARCHAR2,
                segundoApellido VARCHAR2,
                tipo CHAR)
            RETURN SELF AS RESULT IS
        BEGIN
            SELF.codigo := codigo;
            SELF.nombre := nombre;
            SELF.apellidos:= primerApellido || ' ' || segundoApellido;
            SELF.tipo := tipo;
            RETURN;
        END;
        /*3.1 Implementación del Método getNombreCompleto*/
        MEMBER FUNCTION getNombreCompleto RETURN VARCHAR2 IS
        nombreCompleto VARCHAR2(80); -- Declaramos una variable
"nombrecompleto" antes de empezar el bloque
        BEGIN
            nombreCompleto := apellidos || ' ' || nombre; -- Asignamos
a la variable "nombreCompleto" la concatenacion.
            RETURN nombreCompleto; -- Devolvemos  valor de la variable
        END getNombreCompleto; -- Finalizamos el Método
getNombreCompleto.
END;--
```

4. CREA UN TABLA `TablaResponsables` DE OBJETOS `Responsable`. INSERTA EN DICHA TABLA DOS OBJETOS `Responsable`.

```
codigo:  5
nombre: ELENA
apellidos:  POSTA LLANOS
sexo: F
dni: 51083099F
fecha_nac: 31/03/1975
tipo: N
antiguedad: 4
```

```
/*4.CREA UN TABLA TablaResponsables DE OBJETOS  Responsable.*/
-- Creamos o reemplazamos la tabla TablaResponsable con los objetos del
tipo Responsable
CREATE TABLE TablaResponsable OF Responsable;
/ -- Se cierra el bloque
```

```
BEGIN
    /*4.1 INSERTA EN DICHA TABLA DOS OBJETOS  Responsable. */
    -- Insertamos el primer objeto en la tabla TablaResponsable
    INSERT INTO TablaResponsable VALUES (Responsable(5, '51083099F',
'Elena', 'Posta Llanos', 'F', '31-03-1975', 'N', 4));
```

EL SEGUNDO OBJETO "`Responsable`" DEBES CREARLO USANDO EL MÉTODO CONSTRUCTOR QUE HAS REALIZADO ANTERIORMENTE. DEBES USAR LOS SIGUIENTES DATOS:

codigo: 6
nombre: JAVIER
apellidos: JARAMILLO HERNANDEZ
tipo: C

```
    /*4.2 EL SEGUNDO OBJETO "Responsable" DEBES CREARLO USANDO EL
MÉTODO CONSTRUCTOR QUE HAS REALIZADO ANTERIORMENTE. */
    -- Insertamos el segundo objeto utilizando el MÉTODO CONSTRUCTOR.
    -- Para ello, utilizamos "NEW" para llamar a dicho método y luego
lo insertamos en la tabla que acabamos de crear (TablaResponsable).
    INSERT INTO TablaResponsable VALUES (NEW Responsable(6, 'Javier',
'Jaramillo', 'Hernandez', 'C'));
END;
/
```

5. CREA UNA COLECCIÓN `VARRAY` LLAMADA `ListaZonas` EN LA QUE SE PUEDAN ALMACENAR HASTA 10 OBJETOS `Zonas.` . GUARDA EN UNA INSTANCIA `listaZonas1` DE DICHA LISTA, DOS `Zonas`

codigo: 1
nombre: zona 1
refResponsable: Referencia al responsable cuyo codigo es 5
codigo postal: 06834
codigo: 2
nombre: zona 2
refResponsable: Referencia al responsable cuyo DNI es 51083099F.
codigo postal: 28003

```
-- Primero creamos el VARRAY
CREATE TYPE listaZonas IS VARRAY(10) OF Zonas;
/ -- Se cierra el bloque

-- INSTANCIAMOS
DECLARE
```

```
    listaZonas1 listaZonas; -- Declaramos una instancia de la colección
VARRAY "ListasZonas"
    Zona1 Zonas; -- Instanciamos la primera zona del objeto Zonas
    Zona2 Zonas; -- Instanciamos la segunda zona del objeto Zonas

    ref_res REF Responsable; -- Variable que indica la referencia al
tipo de objeto Responsable

BEGIN
    -- Primero: Seleccionamos la referencia de TablaResponsable de la
Zona1
    SELECT REF(r) INTO ref_res FROM TablaResponsable r WHERE r.codigo =
5;
    -- Después, creamos una instancia de Zonas con los valores (Zona1
en este caso)
    Zona1 := NEW Zonas(1, 'Zona 1', ref_res, '06834');

    -- Segundo: Seleccionamos la referencia de Responsable de la Zona2
    SELECT REF(r) INTO ref_res FROM TablaResponsable r WHERE r.dni =
'51083099F';
    -- Creamos una nueva instancia de Zonas con los valores de Zona2
    Zona2 := NEW Zonas(2, 'Zona 2', ref_res, '28003');

    -- Asignamos los objetos Zona1 y Zona2 a la listaZonas1
    listaZonas1 := listaZonas(Zona1, Zona2);
END;

-- Se cierra el bloque anónimo PL/SQL
```

6. CREA UNA TABLA `TablaComerciales` DE OBJETOS `Comercial`. INSERTA EN DICHA TABLA LAS SIGUIENTES FILAS:

codigo: 100
dni: 23401092Z
nombre: MARCOS
apellidos: SUAREZ LOPEZ
sexo: M
fecha_nac: 30/3/1990
zonacomercial: objeto creado anteriormente para la zona 1

codigo: 102
dni: 6932288V
nombre: ANASTASIA

apellidos: GOMES PEREZ
sexo: F
fecha_nac: 28/11/1984
zonacomercial: objeto que se encuentre en la segunda posición de "listaZonas1"
(debe tomarse de la lista)

```
--Creamos la TablaComerciales
CREATE TABLE TablaComerciales OF Comercial;
/

CREATE TYPE listaZonas IS VARRAY(10) OF Zonas;
/

DECLARE
    listaZonas1 listaZonas;
    Zona1 Zonas;
    Zona2 Zonas;

    ref_res REF Responsable;

BEGIN
    SELECT REF(r) INTO ref_res FROM TablaResponsable r WHERE r.codigo =
5;
    Zona1 := NEW Zonas(1, 'Zona 1', ref_res, '06834');
    SELECT REF(r) INTO ref_res FROM TablaResponsable r WHERE r.dni =
'51083099F';
    Zona2 := NEW Zonas(2, 'Zona 2', ref_res, '28003');


    listaZonas1 := listaZonas(Zona1, Zona2);

    -- INSERTAMOS LOS VALORES EN LA "TablaComerciales" con los objetos
de "listaZonas"
    INSERT INTO TablaComerciales VALUES (Comercial(100, '23401092Z',
'MARCOS', 'SUAREZ LOPEZ', 'M', '30-3-1990', listaZonas1(1)));
    INSERT INTO TablaComerciales VALUES (Comercial(102, '6932288V',
'ANASTASIA', 'GOMES PEREZ', 'F', '28-11-1984', listaZonas1(2)));

END;

-- Se cierra el bloque anónimo PL/SQL
```

7. OBTENER, DE LA TABLA `TablaComerciales`, EL `Comercial` QUE TIENE EL CÓDIGO 100, ASIGNÁNDOSELO A UNA VARIABLE `unComercial`

```
    --7.OBTENEMOS EL COMERCIAL CON EL CODIGO 100
    SELECT VALUES (tc) INTO unComercial FROM TablaComerciales tc WHERE
tc.codigo = 100;

END;
```

8. MODIFICA EL CÓDIGO DEL COMERCIAL GUARDADO EN ESA VARIABLE `unComercial` ASIGNANDO EL VALOR 101, Y SU `zona` DEBE SER LA SEGUNDA QUE SE HABÍA CREADO ANTERIORMENTE. INSERTA ESE COMERCIAL EN LA TABLA `TablaComerciales`.

```
CREATE TABLE TablaComerciales OF Comercial;
/

CREATE TYPE listaZonas IS VARRAY(10) OF Zonas;
/

DECLARE
    listaZonas1 listaZonas;
    Zona1 Zonas;
    Zona2 Zonas;

    ref_res REF Responsable;

BEGIN
    SELECT REF(r) INTO ref_res FROM TablaResponsable r WHERE r.codigo =
5;
    Zona1 := NEW Zonas(1, 'Zona 1', ref_res, '06834');
    SELECT REF(r) INTO ref_res FROM TablaResponsable r WHERE r.dni =
'51083099F';
    Zona2 := NEW Zonas(2, 'Zona 2', ref_res, '28003');


    listaZonas1 := listaZonas(Zona1, Zona2);
```

```
    INSERT INTO TablaComerciales VALUES (Comercial(100, '23401092Z',
'MARCOS', 'SUAREZ LOPEZ', 'M', '30-3-1990', listaZonas1(1)));
    INSERT INTO TablaComerciales VALUES (Comercial(102, '6932288V',
'ANASTASIA', 'GOMES PEREZ', 'F', '28-11-1984', listaZonas1(2)));


    SELECT VALUES (tc) INTO unComercial FROM TablaComerciales tc WHERE
tc.codigo = 100;

    /*8. Modifica el código del Comercial guardado en esa variable
unComercial asignando el valor 101,
    y su zona debe ser la segunda que se había creado anteriormente.
Inserta ese Comercial en la tabla TablaComerciales*/
    unComercial.codigo:= 101; -- Por medio del atributo, modificamos el
codigo

    unComercial.ZonaComercial := listaZonas1(2);
    --unComercial es la variable del tipoObjeto Comercial y
ZonaComercial es el atributo de Zonas que también funciona como
atributo de "unComercial"
    --listasZonas1(2) le indicamos que el atributo se tomará del
arreglo listoZonas1, especificamente el 2.

    --Insertamos los valores en la tabla "TablaComerciales"
    INSERT INTO TablaComerciales VALUES (unComercial);

END;
```

9. CREA UN MÉTODO `MAP ordenarZonas` PARA EL TIPO `Zonas`. ESTE MÉTODO DEBE RETORNAR EL NOMBRE COMPLETO DEL `Responsable` AL QUE HACE REFERENCIA CADA `zona.` PARA OBTENER EL NOMBRE DEBES UTILIZAR EL MÉTODO `getNombreCompleto` QUE SE HA CREADO ANTERIORMENTE.

```
CREATE OR REPLACE TYPE Zonas AS OBJECT(
        codigo INTEGER,
        nombre VARCHAR(20),
        refRespon REF Responsable, -- Referimos a objeto "Responsable"
        codigoPostal CHAR(5),

        /*9. Crea un método MAP OrdenarZonas para el tipo Zonas.
```

```
        Este método debe retornar el nombre completo del Responsable
        al que hace referencia cada zona.
        Para obtener el nombre debes utilizar el
         método getNombreCompleto que se ha creado anteriormente*/

        -- 9.1 Creamos Método MAP
        MAP MEMBER FUNCTION OrdenarZonas RETURN VARCHAR2

    );
    /

    --9.2 Creamos el cuerpo del Método MAP OrdenarZonas

    CREATE OR REPLACE TYPE BODY Zonas AS
        MAP MEMBER FUNCTION OrdenarZonas RETURN VARCHAR2 IS
            respon1 Responsable;
        BEGIN
            SELECT DEREF(refRespon) INTO respon1 FROM DUAL; -- Para
navegar y obtener los atributos de un objeto referenciado usamos DEREF
            RETURN respon1.getNombreCompleto();
        END OrdenarZonas;
    END;
    /
```

10. REALIZA UNA CONSULTA DE LA TABLA `TablaComerciales` ORDENADA POR `zonaComercial` PARA COMPROBAR EL FUNCIONAMIENTO DEL MÉTODO `MAP.`

```
/*10.  Realiza una consulta de la tabla TablaComerciales
ordenada por zonaComercial para comprobar el funcionamiento del método
MAP.  */
SELECT * FROM TablaComerciales ORDER BY ZonaComercial;
```

SCRIPT COMPLETO

```
/*TAREA UT07. Uso de bases de datos objeto-relacionales.*/

/* Conforme vamos haciendo los ejercicios, usamos DROP al principio de este script
   para asegurarnos de que no haya objetos existentes que puedan causar conflictos o
errores.
   Esto nos ayuda a mantener un entorno limpio y evitar problemas de dependencia.

   Al colocar las instrucciones DROP en orden de jerarquía, comenzando desde los
objetos
   de menor a mayor nivel de dependencia, garantizamos que los objetos se eliminen
   correctamente y evitamos errores relacionados con dependencias entre objetos. */

DROP TYPE listaZonas;
DROP TABLE TablaComerciales;
DROP TABLE TablaResponsable;
DROP TYPE Comercial;
DROP TYPE Zonas;
DROP TYPE Responsable;
DROP TYPE Persona1; --Tipo Objeto Padre.

/*1.CREA EL TIPO DE OBJETOS "Personal"*/
CREATE OR REPLACE TYPE Persona1 AS OBJECT (
        -- Aquí indicaremos los atributos de nuestro TIPO-OBJETO "Persona1"
        codigo INTEGER,
        dni VARCHAR(10),
        nombre VARCHAR(30),
        apellidos VARCHAR(20),
        sexo VARCHAR(1),
        fecha_nac DATE

        -- Si hubiese métodos, irían aquí.
    ) NOT FINAL; -- Para que otro tipo objeto pueda heredar de "Persona1" debe tener
la propiedad NOT FINAL
    /

/*1.2 CREA, COMO TIPO HEREDADO DE "Personal", EL TIPO DE OBJETO "Responsable" */
-- Especificamos el  método getNombreCompleto dentro  del tipo de objetos
Responsable:
CREATE OR REPLACE TYPE Responsable UNDER Persona1 (
        tipo CHAR,
        antiguedad INTEGER,
```

```
        /* 2.CREA UN MÉTODO CONSTRUCTOR PARA EL TIPO DE OBJETOS "Responsable",
        EN EL QUE SE INDIQUEN COMO PARÁMETROS EL código, nombre, primer apellido,
segundo apellido y tipo.*/
        -- Especificación de atributos del METODO CONSTRUCTOR. Usamos la palabra
CONSTRUCTOR para indicar que es un método constructor.
        CONSTRUCTOR FUNCTION Responsable (
        codigo INTEGER,
        nombre VARCHAR2, -- Usamos VARCHAR2 con el método constructor
        primerApellido VARCHAR2,
        segundoApellido VARCHAR2,
        tipo CHAR
        )
        RETURN SELF AS RESULT,
        /*3. CREA UN MÉTODO getNombreCompleto PARA EL TIPO DE OBJETOS Responsable
        QUE PERMITA OBTENER SU NOMBRE COMPLETO CON EL FORMATO apellidos nombre*/

        -- Especificación del método con MEMBER FUNCTION para que retorne un valor
(MEMBER PROCEDURE no lo hace)
        MEMBER FUNCTION getNombreCompleto RETURN VARCHAR2 -- Usamos "RETURN" para
que nos devuelva el valor.
    );
    /

-- Creamos el cuerpo del tipo Objeto Responsable.
CREATE OR REPLACE TYPE BODY Responsable AS
    CONSTRUCTOR FUNCTION Responsable (
                codigo INTEGER,
                nombre VARCHAR2, -- Usamos VARCHAR2 con el método constructor.
                primerApellido VARCHAR2,
                segundoApellido VARCHAR2,
                tipo CHAR)
            RETURN SELF AS RESULT IS
        BEGIN
        /*2.1 ESTE MÉTODO DEBE ASIGNAR AL ATRIBUTO apellidos LOS DATOS DE PRIMER
APELLIDO Y SEGUNDO APELLIDO QUE SE HAN PASADO COMO PARÁMETROS,
            UNIÉNDOLOS CON UN ESPACIO ENTRE ELLOS*/
            -- Ahora asignamos los valores correspondientes a los atributos usando
SELF
            SELF.codigo := codigo;
            SELF.nombre := nombre;
            SELF.apellidos:= primerApellido || ' ' || segundoApellido; -- Aplicamos
la concatenación en el SELF.apellidos
            SELF.tipo := tipo;
```

```
            RETURN;
        END; -- Finalizamos el bloque del METODO CONSTRUCTOR

        /*3.1 Implementación del Método getNombreCompleto*/
        MEMBER FUNCTION getNombreCompleto RETURN VARCHAR2 IS
        nombreCompleto VARCHAR2(80); -- Declaramos una variable "nombrecompleto"
antes de empezar el bloque
        BEGIN
            nombreCompleto := apellidos || ' ' || nombre; -- Asignamos a la variable
"nombreCompleto" la concatenacion.
            RETURN nombreCompleto; -- Devolvemos  valor de la variable
        END getNombreCompleto; -- Finalizamos el Método getNombreCompleto.
END;--
/


/*1.3 CREA EL TIPO DE OBJETO "Zonas" */
-- Crea tipo de objeto "Zonas" con referencia a objeto "Responsable"
CREATE OR REPLACE TYPE Zonas AS OBJECT(
    codigo INTEGER,
    nombre VARCHAR(20),
    refRespon REF Responsable, -- Referimos a objeto "Responsable"
    codigoPostal CHAR(5),

    /*9. Crea un método MAP OrdenarZonas para el tipo Zonas.
    Este método debe retornar el nombre completo del Responsable
    al que hace referencia cada zona.
    Para obtener el nombre debes utilizar el
        método getNombreCompleto que se ha creado anteriormente*/

    -- 9.1 Creamos Método MAP
    MAP MEMBER FUNCTION OrdenarZonas RETURN VARCHAR2

);
/

/*9.2 Creamos el cuerpo del Método MAP OrdenarZonas.*/
CREATE OR REPLACE TYPE BODY Zonas AS
    MAP MEMBER FUNCTION OrdenarZonas RETURN VARCHAR2 IS
        respon1 Responsable;
    BEGIN
        SELECT DEREF(refRespon) INTO respon1 FROM DUAL; -- Para navegar y obtener
los atributos de un objeto referenciado usamos DEREF
```

```
        RETURN respon1.getNombreCompleto();
    END OrdenarZonas;
END;
/

/*1.4 CREA, COMO TIPO HEREDADO DE "Personal", EL TIPO DE OBJETO "Comercial" */
CREATE OR REPLACE TYPE Comercial UNDER Persona1(
    ZonaComercial Zonas --ZonaComercial es un atributo de tipo "Zonas" (como una
instancia)
);
/



/*4.CREA UN TABLA TablaResponsables DE OBJETOS  Responsable.*/
-- Creamos o reemplazamos la tabla TablaResponsable con los objetos del tipo
Responsable
CREATE TABLE TablaResponsable OF Responsable;
/ -- Se cierra el bloque

BEGIN
    /*4.1 INSERTA EN DICHA TABLA DOS OBJETOS  Responsable. */
    -- Insertamos el primer objeto en la tabla TablaResponsable
    INSERT INTO TablaResponsable VALUES (Responsable(5, '51083099F', 'Elena', 'Posta
Llanos', 'F', '31-03-1975', 'N', 4));

    /*4.2 EL SEGUNDO OBJETO "Responsable" DEBES CREARLO USANDO EL MÉTODO CONSTRUCTOR
QUE HAS REALIZADO ANTERIORMENTE. */
    -- Insertamos el segundo objeto utilizando el MÉTODO CONSTRUCTOR.
    -- Para ello, utilizamos "NEW" para llamar a dicho método y luego lo insertamos
en la tabla que acabamos de crear (TablaResponsable).
    INSERT INTO TablaResponsable VALUES (NEW Responsable(6, 'Javier', 'Jaramillo',
'Hernandez', 'C'));
END;
/



/*6. CREA UNA TABLA TablaComerciales DE OBJETOS Comercial. INSERTA EN DICHA TABLA
LAS SIGUIENTES FILAS:*/
CREATE TABLE TablaComerciales OF Comercial;
/

/*5. CREA UNA COLECCIÓN VARRAY LLAMADA ListaZonas EN LA QUE SE PUEDAN ALMACENAR
HASTA 10 OBJETOS Zonas.*/
```

```
-- Primero creamos el VARRAY
CREATE TYPE listaZonas IS VARRAY(10) OF Zonas;
/

/*5.1 GUARDA EN UNA INSTANCIA listaZonas1 DE DICHA LISTA, DOS Zonas*/
-- INSTANCIAMOS e Iniciamos bloque anónimo
DECLARE
    listaZonas1 listaZonas; -- Declaramos una instancia de la colección VARRAY
"ListasZonas"
    Zona1 Zonas; -- Instanciamos la primera zona del objeto Zonas
    Zona2 Zonas; -- Instanciamos la segunda zona del objeto Zonas
    ref_res REF Responsable; -- Variable que indica la referencia al tipo de objeto
Responsable
    unComercial Comercial; -- Variable para almacenar el Comercial con código 100
BEGIN
    --Primero: Seleccionamos la referencia de TablaResponsable de la Zona1
    SELECT REF(r) INTO ref_res FROM TablaResponsable r WHERE r.codigo = 5;
    -- Después, creamos una instancia de Zonas con los valores (Zona1 en este caso)
    Zona1 := NEW Zonas(1, 'Zona 1', ref_res, '06834');

    --Segundo: Seleccionamos la referencia de Responsable de la Zona2
    SELECT REF(r) INTO ref_res FROM TablaResponsable r WHERE r.dni = '51083099F';
    -- Creamos una nueva instancia de Zonas con los valores de Zona2
    Zona2 := NEW Zonas(2, 'Zona 2', ref_res, '28003');

    -- Asignamos los objetos Zona1 y Zona2 a la listaZonas1
    listaZonas1 := listaZonas(Zona1, Zona2);

    /*6.1 INSERTAMOS LOS VALORES EN LA "TablaComerciales" con los objetos de
"listaZonas"*/
    INSERT INTO TablaComerciales VALUES (Comercial(100, '23401092Z', 'MARCOS',
'SUAREZ LOPEZ', 'M', '30-3-1990', listaZonas1(1)));
    INSERT INTO TablaComerciales VALUES (Comercial(102, '6932288V', 'ANASTASIA',
'GOMES PEREZ', 'F', '28-11-1984', listaZonas1(2)));

    /*7. Obtener el Comercial con código 100 y asignarlo a la variable unComercial*/
    SELECT VALUE(tc) INTO unComercial FROM TablaComerciales tc WHERE tc.codigo =
100;

    /*8. Modifica el código del Comercial guardado en esa variable unComercial
asignando el valor 101,
    y su zona debe ser la segunda que se había creado anteriormente. Inserta ese
Comercial en la tabla TablaComerciales*/
```

```
    unComercial.codigo:= 101; -- Por medio del atributo, modificamos el codigo

    unComercial.ZonaComercial := listaZonas1(2);
    --unComercial es la variable del tipoObjeto Comercial y ZonaComercial es el
atributo de Zonas que también funciona como atributo de "unComercial"
    --listasZonas1(2) le indicamos que el atributo se tomará del arreglo
listoZonas1, especificamente el 2.

    /*8.1 Insertamos los valores en la tabla "TablaComerciales"*/
    INSERT INTO TablaComerciales VALUES (unComercial);
END;
/
-- Se cierra el bloque anónimo

/*10.  Realiza una consulta de la tabla TablaComerciales
ordenada por zonaComercial para comprobar el funcionamiento del método MAP.  */
SELECT * FROM TablaComerciales ORDER BY ZonaComercial;
```