Acceso a Datos

Tarea para AD05.

Enunciado.

EJERCICIO 1

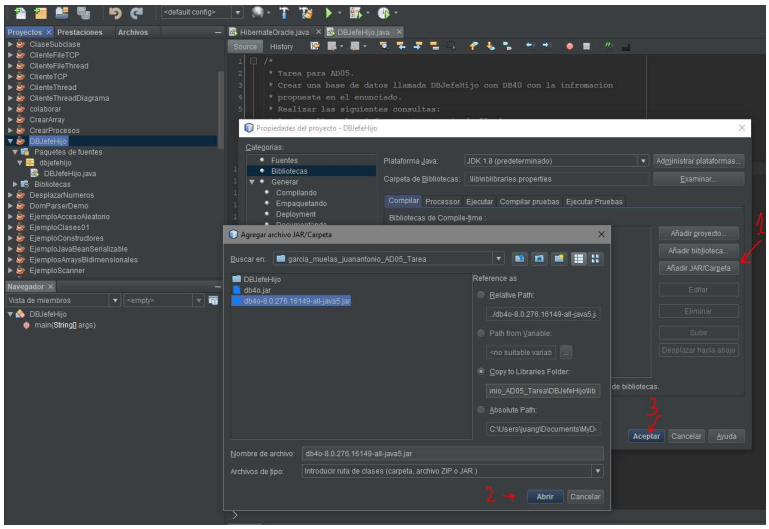
Crear una base de datos llamada DBJefeHijo con DB40 con la siguiente información:

```
public static void main(String[] args)
{
File fichero=new File("baseDatos");
fichero.delete();
/*Este código anterior lo ponemos por si la base de datos ya existiera y quisiéramos empezar desde el principio.*/
ObjectContainer baseDatos=Db4oEmbedded.openFile("BDJefeHijo ");
baseDatos.store(new Jefe("Ángel", 5, 53,new Hijo("Gustavo", 7)));
baseDatos.store(new Jefe("Nieves", 3, 45,new Hijo("Iván", 3)));
baseDatos.store(new Jefe("Jesús", 3, 5,new Hijo("Noelia", 3)));
baseDatos.store(new Jefe("Dolores", 5,63,new Hijo("Sergio", 7)));
baseDatos.store(new Jefe("Vicki", 3, 5,null));
baseDatos.store(new Jefe("Fátima", 5,63,new Hijo("Lidia", 27)));
baseDatos.store(new Jefe("Juan Luís", 3, 5,null));
baseDatos.store(new Jefe("Elena", 1,42,new Hijo("David", 19)));
baseDatos.store(new Jefe("Miguel", 20,45,new Hijo("Paula", 3)));
baseDatos.store(new Jefe("Jesús", 19, 44,new Hijo("Rubén", 12)));
baseDatos.close();
}
```

Realizar las siguientes consultas:

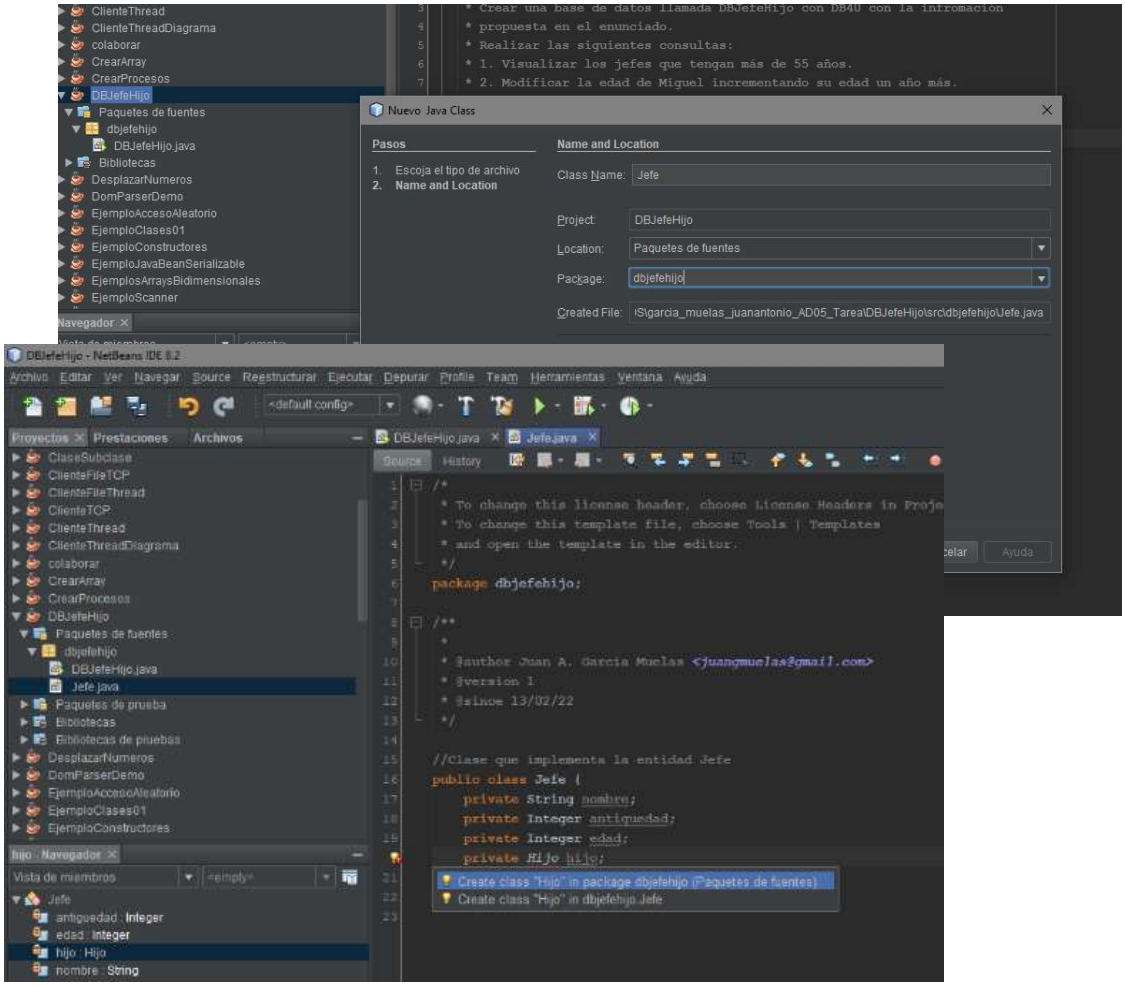
1. Visualizar los jefes que tengan más de 55 años.
2. Modificar la edad de Miguel incrementando su edad un año más.
3. Borrar los jefes que llevan más de 6 años en la empresa.
4. Visualizar todos los jefes que quedan, incluidos sus hijos, que no han sido borrados anteriormente.

Para realizar esta tarea lo primero que hago es crear un nuevo proyecto Java, que voy a llamar como la base de datos propuesta para el ejercicio.



Tras añadir el `jar` de conexión, siguiendo los ejemplos facilitados para su descarga en el temario, vamos a crear las clases `Jefe` e `Hijo` que utilizaremos para manejar los objetos necesarios en los ejercicios con nuestra base de datos.

Se deduce por el código facilitado que van a almacenarse objetos de tipo jefe, mediante el método `store()`, con los siguientes campos para `Jefe` (nombre, antigüedad, edad e hijo) y para `Hijo` (nombre, edad). Con estos datos creamos los archivos con sus propiedades y métodos de acceso.



Siguiendo la costumbre de anteriores ejercicios, paso a mostrar el código de las clases que componen su resolución, confiando sean suficientemente ilustrativas.

Jefe.java

```
/*
 * Tarea para AD05.
 * Crear una base de datos llamada DBJefeHijo con DB40 con la
 * información propuesta en el enunciado.
 * Realizar las siguientes consultas:
 * 1. Visualizar los jefes que tengan más de 55 años.
 * 2. Modificar la edad de Miguel incrementando su edad un año más.
 * 3. Borrar los jefes que llevan más de 6 años en la empresa.
 * 4. Visualizar todos los jefes que quedan, incluidos sus hijos,
 *    que no han sido borrados anteriormente.
 */

package dbjefehijo;
```

```java
/**
 *
 * @author Juan A. García Muelas <juangmuelas@gmail.com>
 * @version 1
 * @since 13/02/22
 */

//Clase que implementa la entidad Jefe
public class Jefe {
    /**
     * @param String nombre variable de tipo texto recoge nombre.
     * @param Integer antiguedad variable de tipo entero para
     * recoger años en la empresa.
     * @param Integer edad variable de tipo entero para edad.
     * @param Hijo hijo objeto clase tipo Hijo.
     */
    private String nombre;
    private Integer antiguedad;
    private Integer edad;
    private Hijo hijo;

    //Constructores
    public Jefe() {
    }

    public Jefe(String nombre, Integer antiguedad, Integer edad,
    Hijo  hijo) {
        this.nombre = nombre;
        this.antiguedad = antiguedad;
        this.edad = edad;
        this.hijo = hijo;
    }

    //Getter y setters
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public Integer getAntiguedad() {
        return antiguedad;
    }

    public void setAntiguedad(Integer antiguedad) {
        this.antiguedad = antiguedad;
    }

    public Integer getEdad() {
        return edad;
    }

    public void setEdad(Integer edad) {
        this.edad = edad;
    }

    public Hijo getHijo() {
        return hijo;
    }
```

```
    public void setHijo(Hijo hijo) {
        this.hijo = hijo;
    }

    @Override

    /**
     * Comportamiento del método toString heredado de la superclase
     * Object que devuelve los atributos de un objeto Jefe.
     */
    public String toString() {
        return "Jefe{" + "nombre=" + nombre + ", antiguedad="
             + antiguedad + ", edad=" + edad + ", hijo=" + hijo + '}';
    }

}//fin clase Jefe.java
```

Hijo.java

```
/*
 * Tarea para AD05.
 * Crear una base de datos llamada DBJefeHijo con DB40 con la
 * información propuesta en el enunciado.
 * Realizar las siguientes consultas:
 * 1. Visualizar los jefes que tengan más de 55 años.
 * 2. Modificar la edad de Miguel incrementando su edad un año más.
 * 3. Borrar los jefes que llevan más de 6 años en la empresa.
 * 4. Visualizar todos los jefes que quedan, incluidos sus hijos,
 *    que no han sido borrados anteriormente.
 */
package dbjefehijo;

/**
 *
 * @author Juan A. García Muelas <juangmuelas@gmail.com>
 * @version 1
 * @since 13/02/22
 */
public class Hijo {
    /**
     * @param String nombre variable de tipo texto recoge nombre.
     * @param Integer edad variable de tipo entero para edad.
     */
    private String nombre;
    private Integer edad;

    //constructores
    public Hijo(String nombre, Integer edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    //getters y setters
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
```

```
    public Integer getEdad() {
        return edad;
    }

    public void setEdad(Integer edad) {
        this.edad = edad;
    }

    @Override
    public String toString() {
        return "Hijo{" + "nombre=" + nombre + ", edad=" + edad + '}';
    }

}//fin clase Hijo
```

DBJefeHijo.java

Las dos primeras clases, no necesitan de mucha más explicación tras todo lo estudiado anteriormente, pero si quiero detallar algunos aspectos de la clase principal.

Aunque en el enunciado se incluye la inserción de datos a fichero desde el método `main`, he creído, siguiendo los ejemplos facilitados para manejar objetos estructurados en el temario, que era positivo añadir toda esa lógica en una nueva clase (`ConexionDb4o`), que por no complicar más el directorio de proyecto, he añadido en el mismo archivo. Con ella, creamos un objeto que nos permitirá trabajar y mostrar datos en el `main`.

Los métodos de acceso a datos, están en buena parte recogidos en los apuntes y ejemplos del temario, por lo que nos pueden sonar familiares incluso algunos comentarios, que me ha parecido razonable mantener para dar una mejor comprensión al ejercicio.

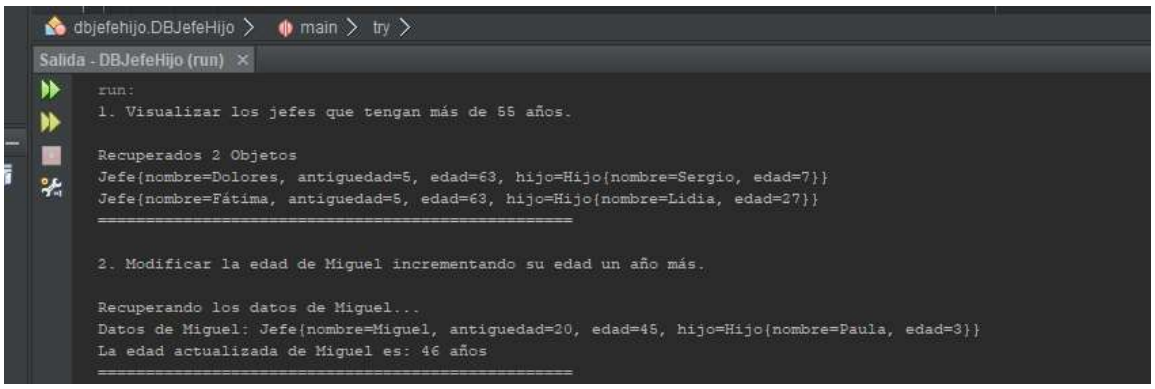
```
/*
 * Tarea para AD05.
 * Crear una base de datos llamada DBJefeHijo con DB40 con la
 * información propuesta en el enunciado.
 * Realizar las siguientes consultas:
 * 1. Visualizar los jefes que tengan más de 55 años.
 * 2. Modificar la edad de Miguel incrementando su edad un año más.
 * 3. Borrar los jefes que llevan más de 6 años en la empresa.
 * 4. Visualizar todos los jefes que quedan, incluidos sus hijos,
 *    que no han sido borrados anteriormente.
 */
package dbjefehijo;

import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
import com.db4o.config.EmbeddedConfiguration;
import com.db4o.query.Query;
import java.io.File;

/**
 *
 * @author Juan A. García Muelas <juangmuelas@gmail.com>
 * @version 1
 * @since 13/02/22
 */
class ConexionDb4o{

    private ObjectContainer baseDatos = miDb4o();
```

```
    public ObjectContainer miDb4o() {
        File fichero=new File("baseDatos");
        fichero.delete();
        /* Este código anterior lo ponemos por si la base de datos
         * ya existiera y quisiéramos empezar desde el principio.
         */

        EmbeddedConfiguration config =
        Db4oEmbedded.newConfiguration();
        config.common().objectClass(Jefe.class).cascadeOnDelete(true);
        try {
            if (baseDatos == null) {
                ObjectContainer baseDatos =
                Db4oEmbedded.openFile("BDJefeHijo ");
                baseDatos.store(new Jefe("Ángel", 5, 53, new
                Hijo("Gustavo", 7)));
                baseDatos.store(new Jefe("Nieves", 3, 45, new
                Hijo("Iván", 3)));
                baseDatos.store(new Jefe("Jesús", 3, 5, new
                Hijo("Noelia", 3)));
                baseDatos.store(new Jefe("Dolores", 5, 63, new
                Hijo("Sergio", 7)));
                baseDatos.store(new Jefe("Vicki", 3, 5, null));
                baseDatos.store(new Jefe("Fátima", 5, 63, new
                Hijo("Lidia", 27)));
                baseDatos.store(new Jefe("Juan Luís", 3, 5, null));
                baseDatos.store(new Jefe("Elena", 1, 42, new
                Hijo("David", 19)));
                baseDatos.store(new Jefe("Miguel", 20, 45, new
                Hijo("Paula", 3)));
                baseDatos.store(new Jefe("Jesús", 19, 44, new
                Hijo("Rubén", 12)));
                baseDatos.query();

                return baseDatos;
            }
        } catch (Exception e) {
            if(baseDatos != null){
                baseDatos.close();
                //cerrar base de datos antes de  salir
            }
        }
        return this.baseDatos;
    }//fin método miDb40

    //Método para mostrar objetos recuperados de la Base de Objetos
    public void mostrarConsulta(ObjectSet resul) {
        //mensaje indicando el total de objetos recuperados
        System.out.println("Recuperados " + resul.size()
                + " Objetos");
        while (resul.hasNext()) {//bucle que obtiene objeto a objeto
            System.out.println(resul.next());
        }
    }


    // 1. Visualizar los jefes que tengan más de 55 años.
    //Consulta SODA de objetos estructurados
    //Se consulta el jefe cuya edad se pasa en la restricción
    public void jefesMas55() {
        Query query = baseDatos.query();//declara un objeto query
```

```
        query.constrain(Jefe.class);//establece la clase a la que se
                                    aplicará la restricción
        query.descend("edad").constrain(55).greater();//establece la
                                                restricción de búsqueda
        ObjectSet result = query.execute();//ejecuta consulta
        mostrarConsulta(result);//método que muestra los objetos
                                recuperados de la BDOO
    }

    // 2. Modificar la edad de Miguel incrementando edad un año más.
    //Modificación de Objetos estructurados. Con consulta QBE
    public void modificarEdad(String nombre) {
        //consulta Jefe por nombre.Consulta QBE
        ObjectSet res = baseDatos.queryByExample(new Jefe(nombre,
                        null, null, null));
        Jefe jefe = (Jefe) res.next();//obtiene el jefe consultado
        //mostramos datos
        System.out.printf("Datos de %s: %s\n", nombre, jefe);

        jefe.setEdad(jefe.getEdad() + 1);//cogemos edad aumentando 1
        baseDatos.store(jefe);//almacena jefe modificado y repetimos
                              proceso
        res = baseDatos.queryByExample(new Jefe(nombre, null, null,
                                                null));
        jefe = (Jefe) res.next();//obtenido el jefe se muestra.
        System.out.printf("La edad actualizada de %s es: %s años \n",
                          jefe.getNombre(), jefe.getEdad());
    }

    // 3. Borrar los jefes que llevan más de 6 años en la empresa.
    // Borrado de objetos estructurados. Se utiliza Consulta SODA
    // recordar la estructura del punto 1
    public void borrarJefesMas6() {
        Query query = baseDatos.query();
        query.constrain(Jefe.class);
        query.descend("antiguedad").constrain(6).greater();
        ObjectSet resul = query.execute();
        //bucle que recupera los objetos jefe y elimina de la BDOO
        while (resul.hasNext()) {
            Jefe jefe = (Jefe) resul.next();
            System.out.printf("Borrando a: %s\n", jefe);
            baseDatos.delete(jefe);
        }
    }

    // 4. Visualizar todos los jefes que quedan, incluidos sus hijos,
    // que no han sido borrados anteriormente.
    // Consulta SODA similar al punto 1, pero sin restricciones.
    public void mostrarTodos(){
        Query query = baseDatos.query();
        query.constrain(Jefe.class);
        ObjectSet result = query.execute();
        mostrarConsulta(result);
    }

    // Finalizamos conexiones.
    public void cerrarConexion() {
        baseDatos.commit();
        baseDatos.close();
    }
}//Fin clase ConexionDB4o
```

```
public class DBJefeHijo {
    public static void main(String[] args) {
        ConexionDb4o connDb4o = null;
        try {
            connDb4o = new ConexionDb4o();
            //Mostramos la primera consulta
            System.out.println("1. Visualizar los jefes que tengan más
                                de 55 años.\n");
            connDb4o.jefesMas55();

            System.out.println("================================\n");

            //Mostramos la segunda consulta
            String nombre = "Miguel";
            System.out.printf("2. Modificar la edad de %s
                    incrementando su edad un año más.\n\n", nombre);
            System.out.printf("Recuperando los datos de %s...\n",
                               nombre);
            connDb4o.modificarEdad(nombre);
            System.out.println("================================\n");

            //tercera consulta
            System.out.println("3. Borrar los jefes que llevan más de
                                6 años en la empresa.\n");
            connDb4o.borrarJefesMas6();
            System.out.println("================================\n");

            //Cuarta y última consulta de este ejercicio
            System.out.println("4. Visualizar todos los jefes que
                                quedan, "
                              + "incluidos sus hijos, que no han sido
                                borrados anteriormente.\n");
            connDb4o.mostrarTodos();
        } catch (Exception e) {
            System.err.print("Error al ejecutar la consulta!");
            e.printStackTrace();
        } finally {
            if (connDb4o != null) {
                connDb4o.cerrarConexion();
            }
        }
    }  //fin main
}//Fin clase DBJefeHijo.java
```

Si ejecutamos, podremos ver los puntos solicitados en la consola:



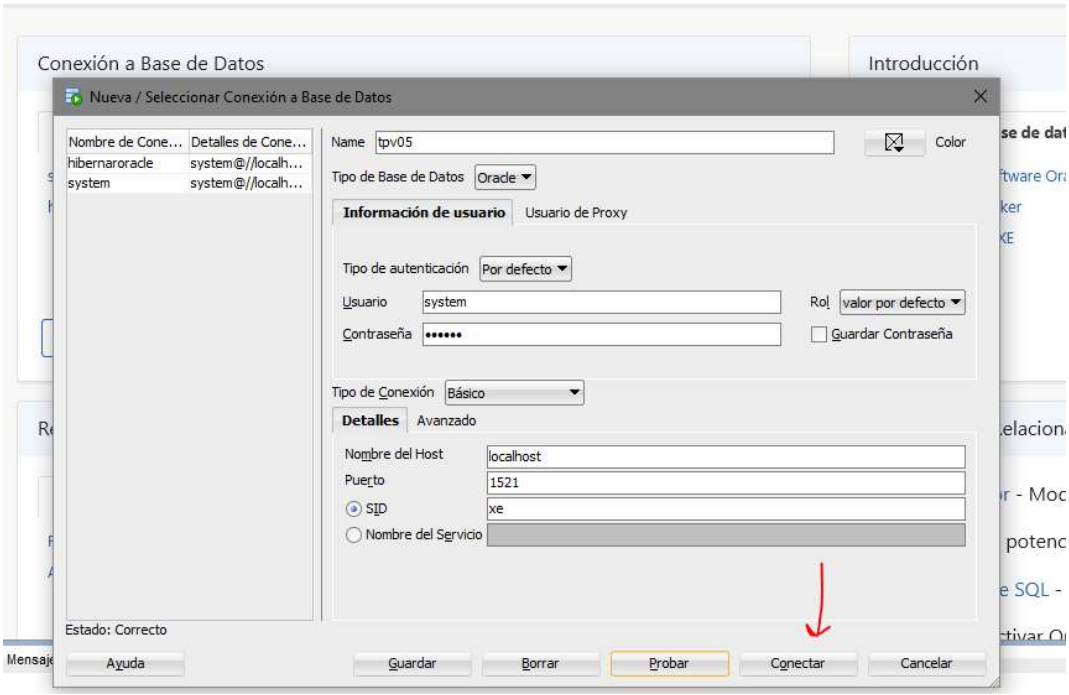
```
3. Borrar los jefes que llevan más de 6 años en la empresa.

Borrando a: Jefe{nombre=Miguel, antiguedad=20, edad=46, hijo=Hijo{nombre=Paula, edad=3}}
Borrando a: Jefe{nombre=Jesús, antiguedad=19, edad=44, hijo=Hijo{nombre=Rubén, edad=12}}
==================================================

4. Visualizar todos los jefes que quedan, incluidos sus hijos, que no han sido borrados anteriormente.

Recuperados 8 Objetos
Jefe{nombre=Ángel, antiguedad=5, edad=53, hijo=Hijo{nombre=Gustavo, edad=7}}
Jefe{nombre=Nieves, antiguedad=3, edad=45, hijo=Hijo{nombre=Iván, edad=3}}
Jefe{nombre=Jesús, antiguedad=3, edad=5, hijo=Hijo{nombre=Noelia, edad=3}}
Jefe{nombre=Dolores, antiguedad=5, edad=63, hijo=Hijo{nombre=Sergio, edad=7}}
Jefe{nombre=Vicki, antiguedad=3, edad=5, hijo=null}
Jefe{nombre=Fátima, antiguedad=5, edad=63, hijo=Hijo{nombre=Lidia, edad=27}}
Jefe{nombre=Juan Luis, antiguedad=3, edad=5, hijo=null}
Jefe{nombre=Elena, antiguedad=1, edad=42, hijo=Hijo{nombre=David, edad=19}}
BUILD SUCCESSFUL (total time: 0 seconds)
```

EJERCICIO 2

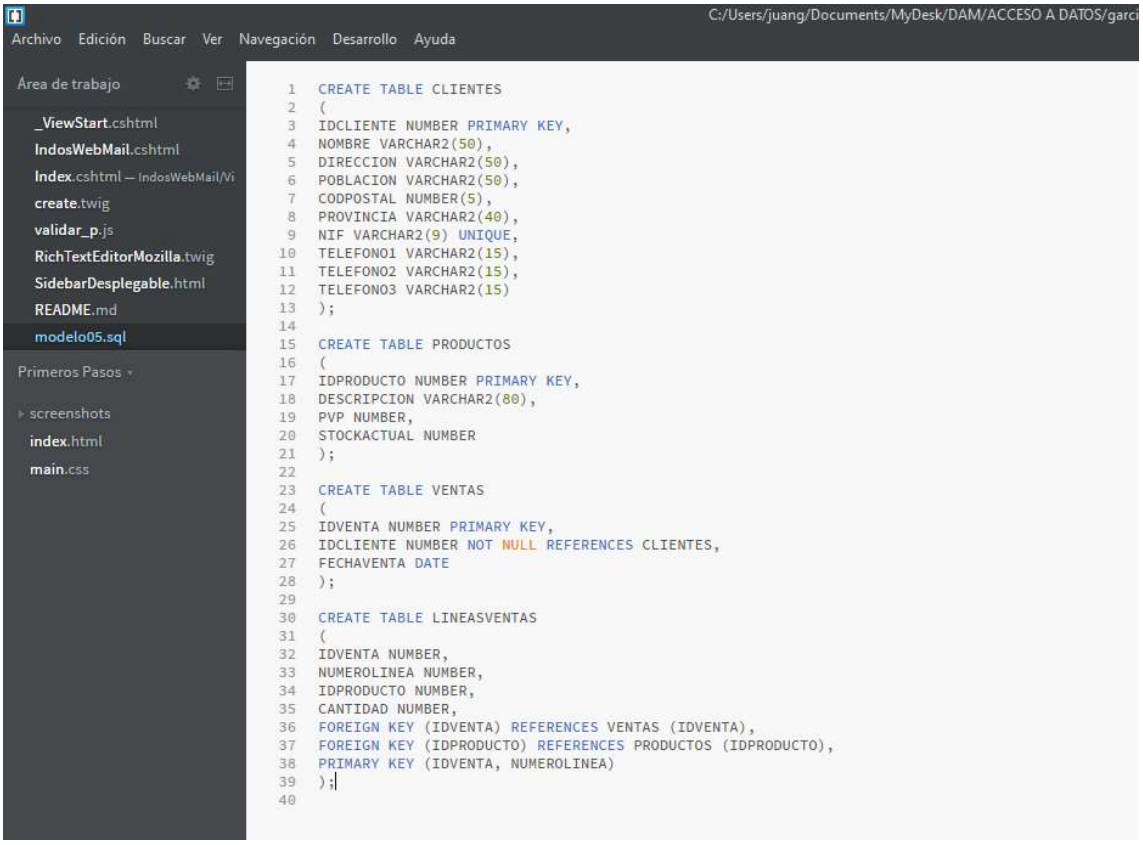
Dado el siguiente modelo de datos

```
CREATE TABLE CLIENTES
(
IDCLIENTE NUMBER PRIMARY KEY,
NOMBRE VARCHAR2(50),
DIRECCION VARCHAR2(50),
POBLACION VARCHAR2(50),
CODPOSTAL NUMBER(5),
PROVINCIA VARCHAR2(40),
NIF VARCHAR2(9) UNIQUE,
TELEFONO1 VARCHAR2(15),
TELEFONO2 VARCHAR2(15),
TELEFONO3 VARCHAR2(15)
);

CREATE TABLE PRODUCTOS
(
IDPRODUCTO NUMBER PRIMARY KEY,
DESCRIPCION VARCHAR2(80),
PVP NUMBER,
STOCKACTUAL NUMBER
);

CREATE TABLE VENTAS
(
IDVENTA NUMBER PRIMARY KEY,
IDCLIENTE NUMBER NOT NULL REFERENCES CLIENTES,
FECHAVENTA DATE
);

CREATE TABLE LINEASVENTAS
(
IDVENTA NUMBER,
NUMEROLINEA NUMBER,
IDPRODUCTO NUMBER,
CANTIDAD NUMBER,
FOREIGN KEY (IDVENTA) REFERENCES VENTAS (IDVENTA),
FOREIGN KEY (IDPRODUCTO) REFERENCES PRODUCTOS (IDPRODUCTO),
PRIMARY KEY (IDVENTA, NUMEROLINEA)
);
```

Accedo a SQLDeveloper y creo una nueva conexión llamada tpv05(por dar un nombre asociado a la posible finalidad de las tablas propuestas y la tarea)

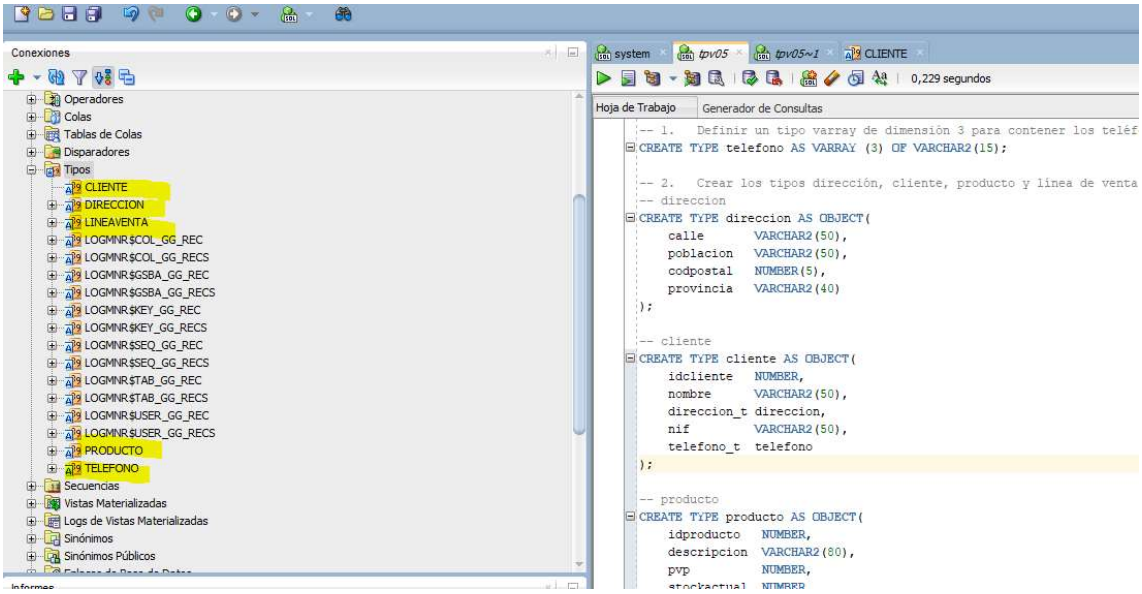


Guardo el modelo propuesto en un archivo sql y con él, creo las tablas.

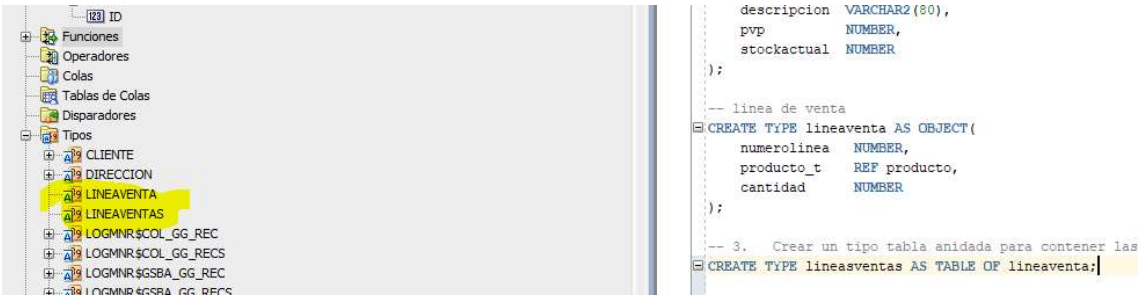


Para mostrar la realización de esta parte, intentaré que en las capturas se muestre de la mejor forma posible, las distintas consultas generadas a la base de datos, que de todas formas , dejaré en documento aparte para su comprobación y uso.

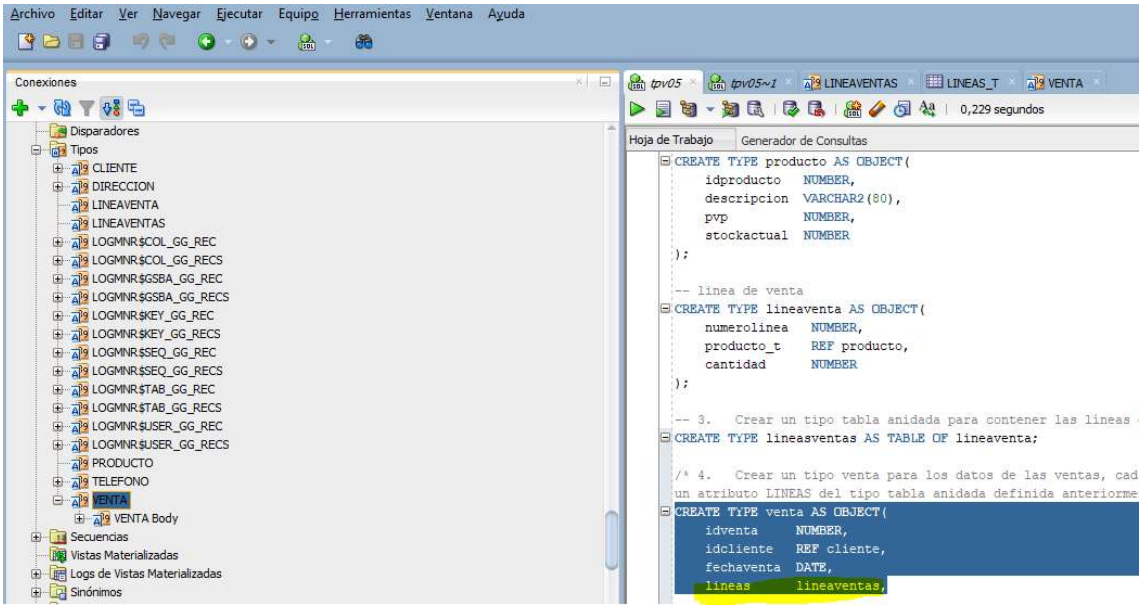
1. **Definir un tipo varray de dimensión 3 para contener los teléfonos.**
2. **Crear los tipos dirección, cliente, producto y línea de venta.**



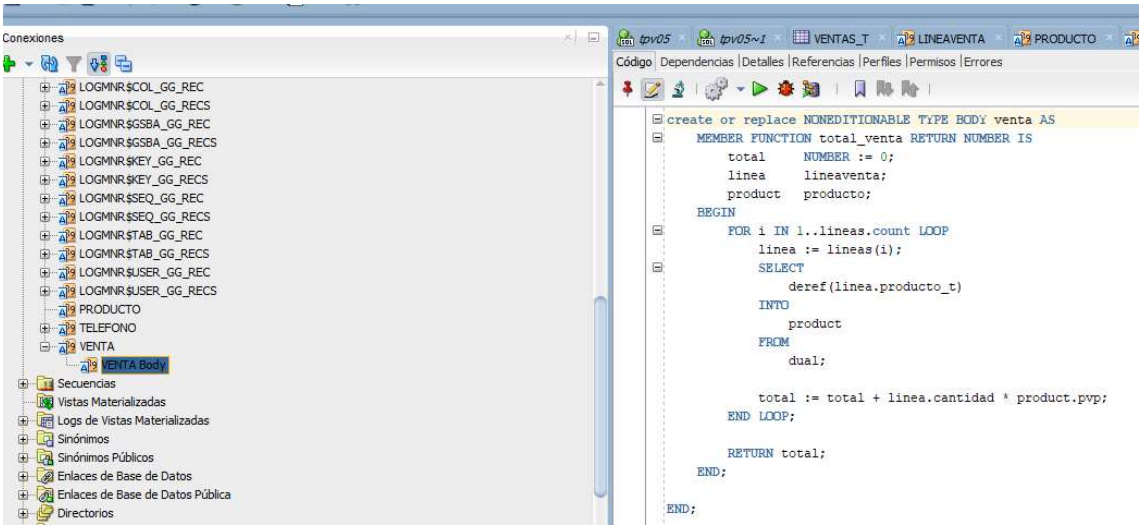
3. **Crear un tipo tabla anidada para contener las líneas de una venta:**



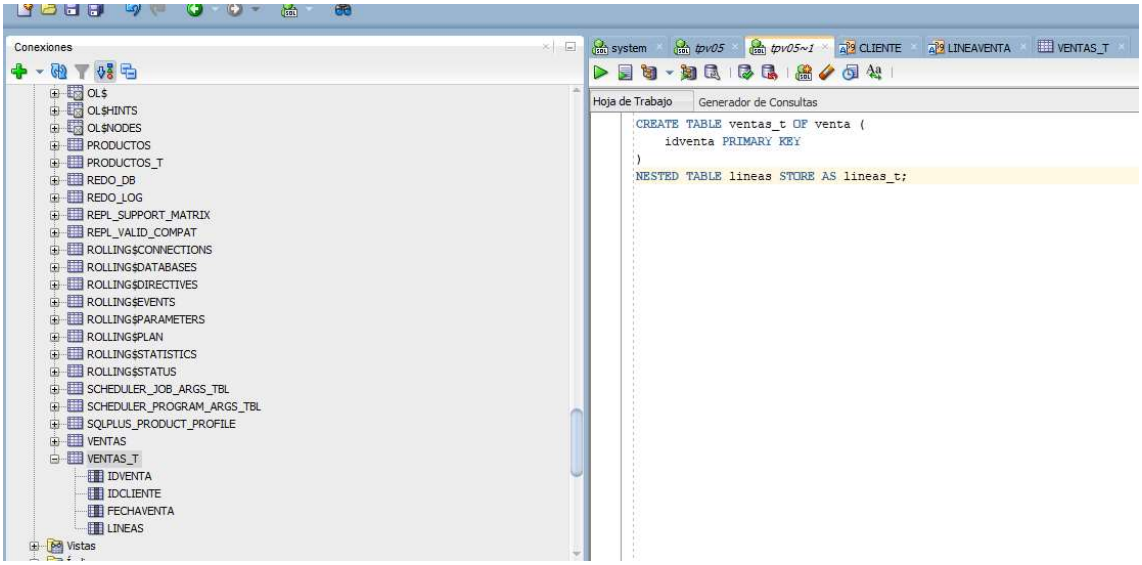
4. **Crear un tipo venta para los datos de las ventas, cada venta tendrá un atributo LINEAS del tipo tabla anidada definida anteriormente:**



5. **Crea el cuerpo del tipo anterior, teniendo en cuenta que se definirá la función miembro TOTAL_VENTA que calcula el total de la venta de las líneas de venta que forman parte de una venta, contará el número de elementos de una tabla o de un array y devolverá el número de líneas que tiene la venta.**

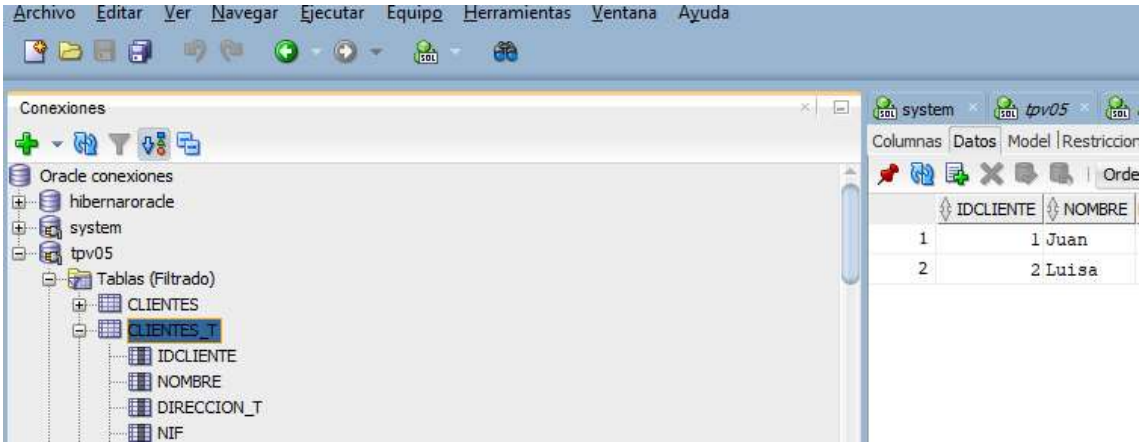


6. **Crear las tablas donde almacenar los objetos de la aplicación. Se creará una tabla para clientes, otra para productos y otra para las ventas, en dichas tablas se definirán las oportunas claves primarias.**

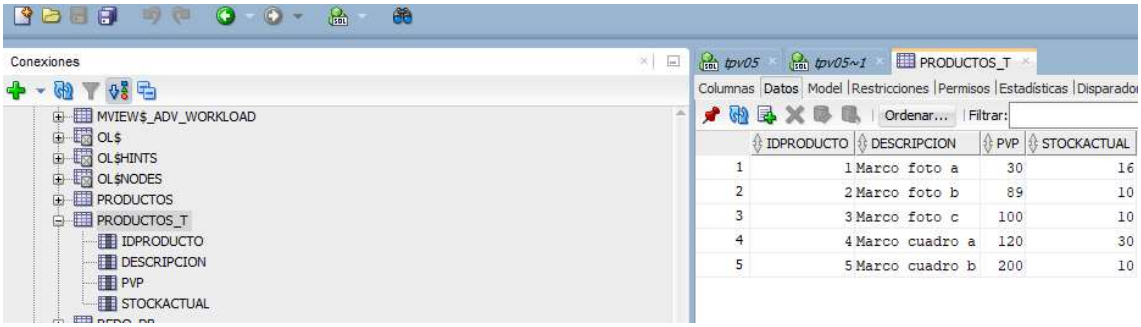


7. **Inserta dos clientes y cinco productos.**

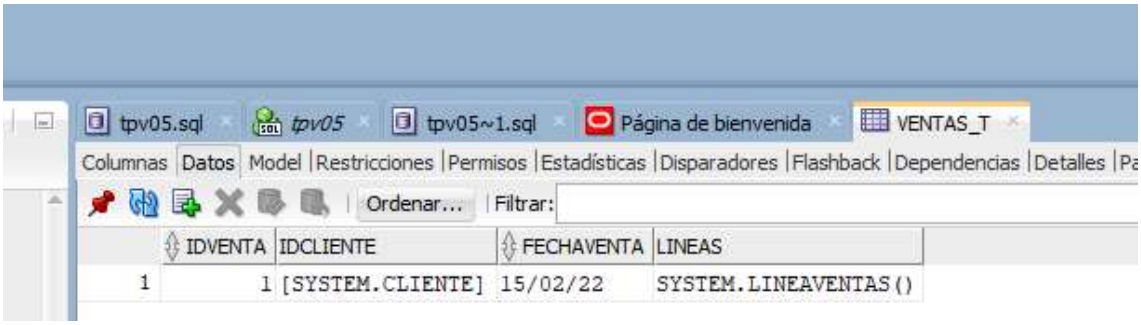
Clientes



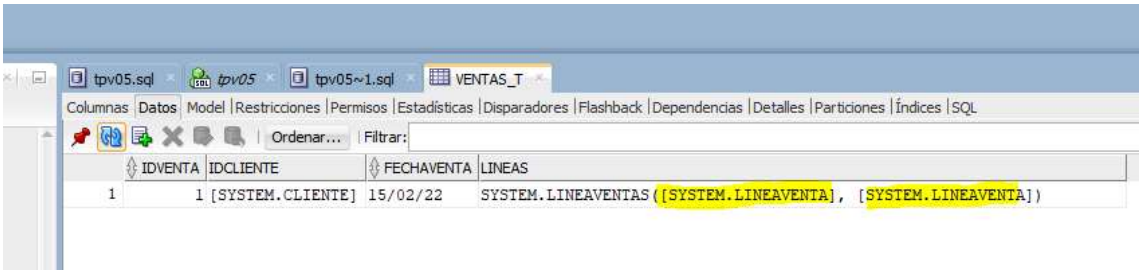
Productos



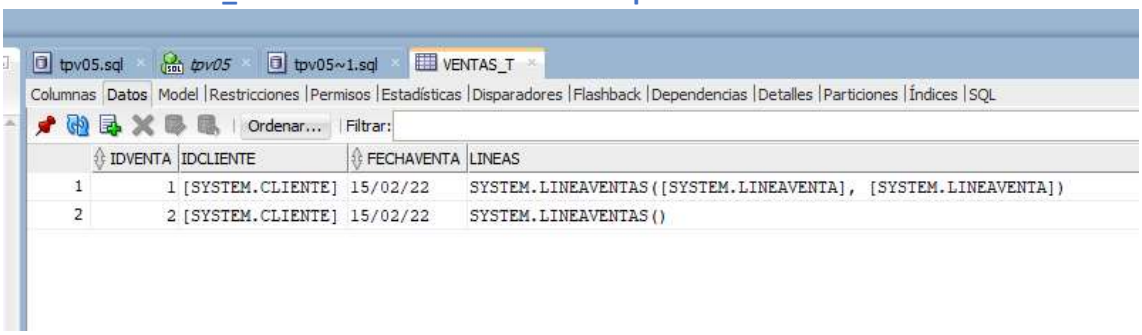
8. **Insertar en TABLA_VENTAS la venta con IDVENTA 1 para el IDCLIENTE 1**



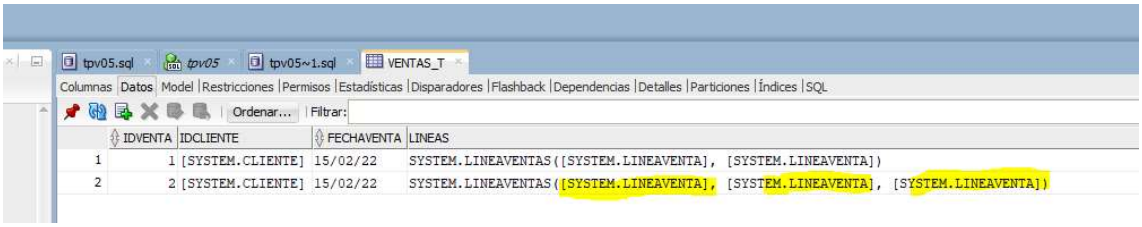
9. **Insertar en TABLA_VENTAS dos líneas de venta para el IDVENTA 1 para los productos 1 (la CANTIDAD es 1) y 2 (la CANTIDAD es 2)**



10. **Insertar en TABLA_VENTAS la venta con IDVENTA 2 para el IDCLIENTE**

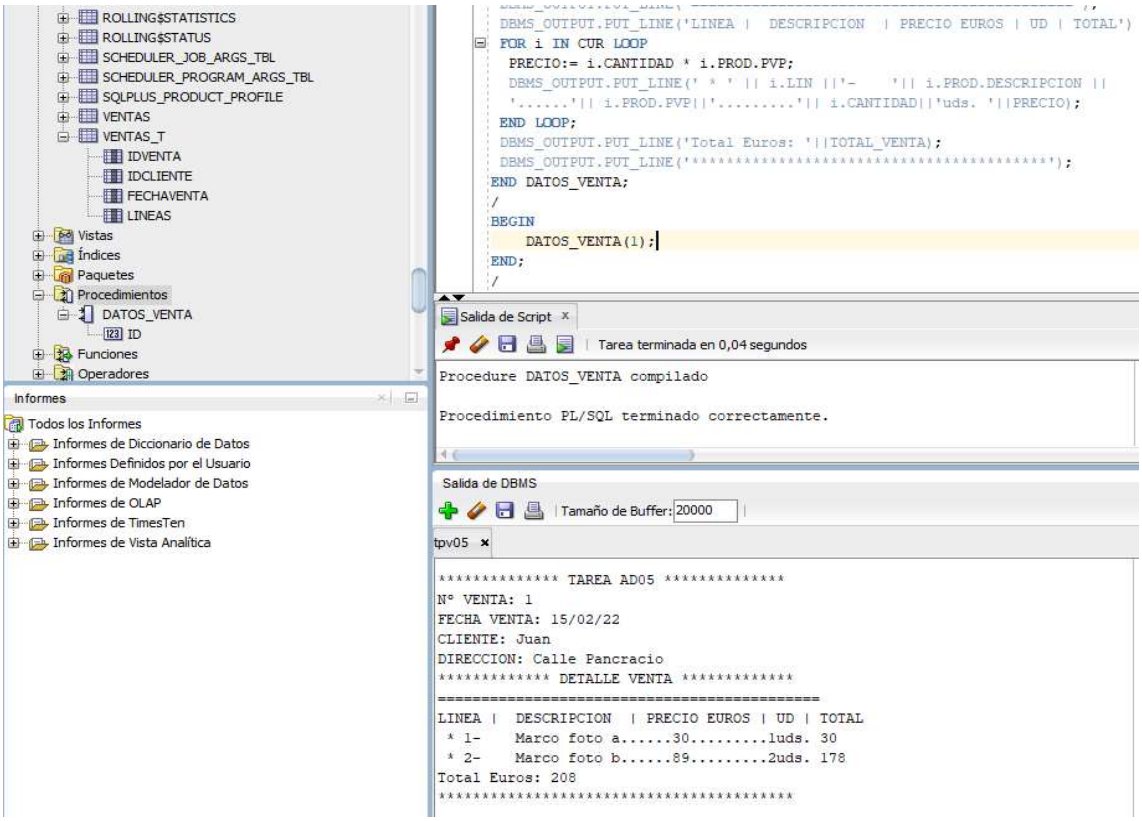


11. **Insertar en TABLA_VENTAS tres líneas de venta para el IDVENTA 2 para los productos 1 (la CANTIDAD es 2), 4 (la CANTIDAD es 1) y 5 (la CANTIDAD es 4)**



12. **Realizar un procedimiento que recibiendo el identificador visualice los datos de la venta. En `Vista`, activamos la `Salida de DBMS` para nuestra conexión y ejecutamos el procedimiento.**

Idventa 1



Idventa 2