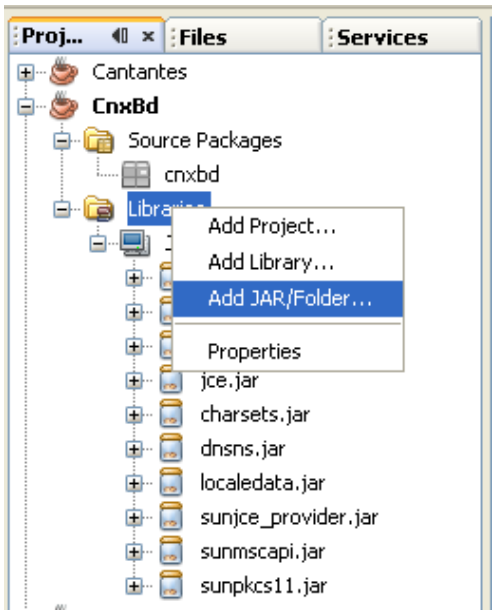
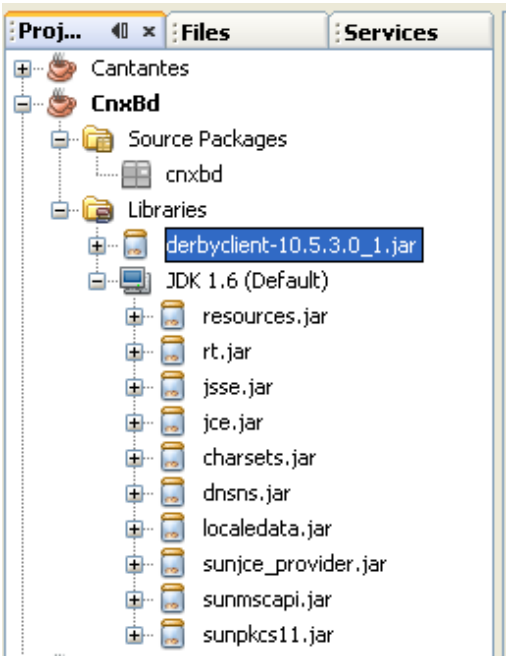
UNIVERSIDAD DISTRITAL FRANCSICO JOSE DE CALDAS
FACULTAD DE INGENIERIA - INGENIERIA DE SISTEMAS
PROGRAMACION AVANZADA

TALLER DE CONEXIÓN A BASES DE DATOS

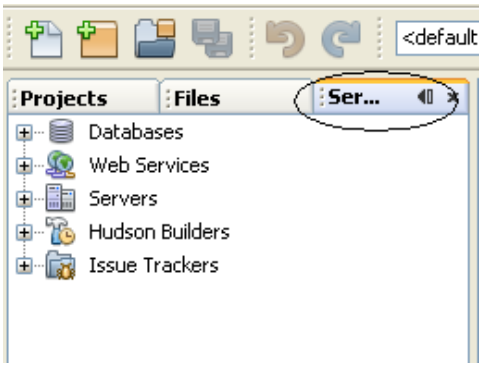
1. Cree un nuevo proyecto Java Aplication
2. Descargue el respectivo driver para conectarse a una BD Derby. El driver JDBC para Derby: derbyclient.jar
3. Agregue el driver a la carpeta de librerías



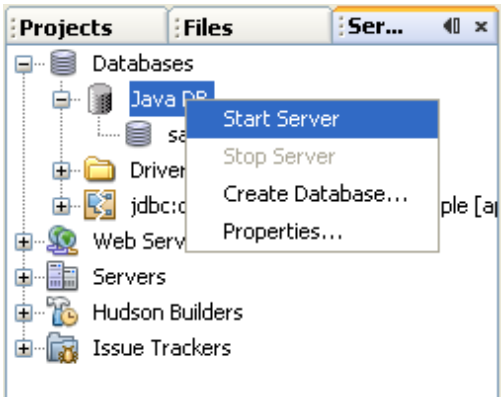
4. Después de adjuntarlo, debe aparecer como una jar o librería adicional



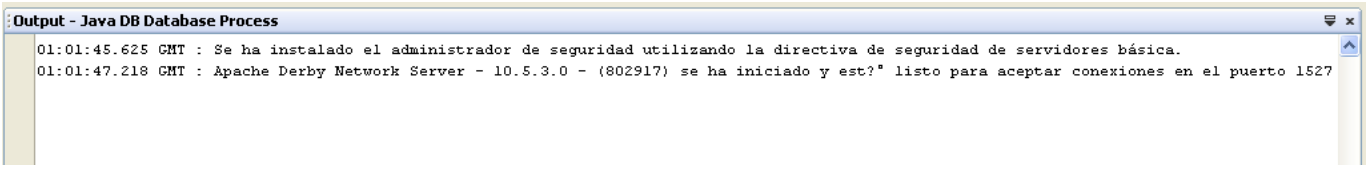
5. A continuación, inicialice el servicio de **Derby**:
   a. En la pestaña de servicios



   b. Inicie el servidor de Bases de Datos

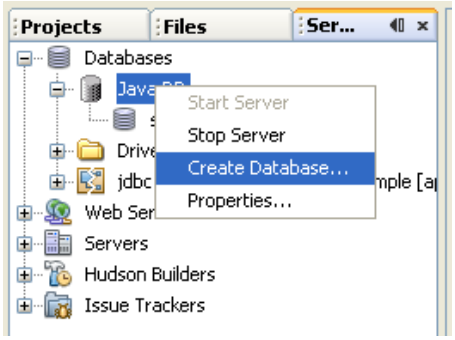


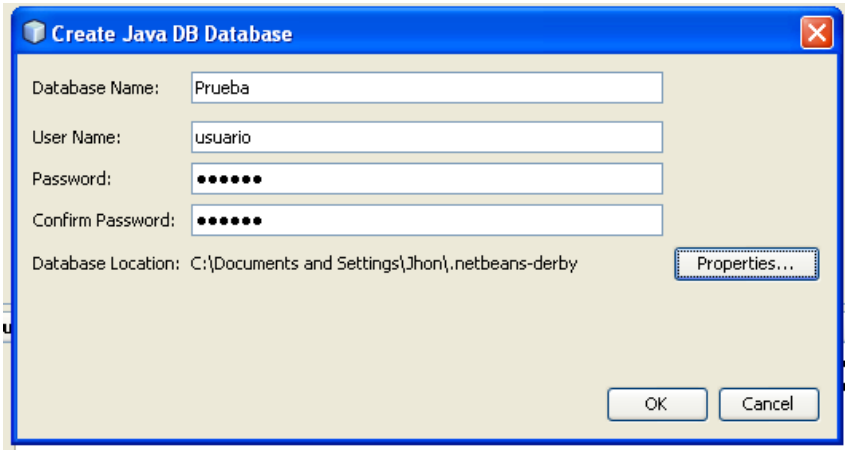
   c. Tan pronto se inicie el servidor, en la consola aparecerá:

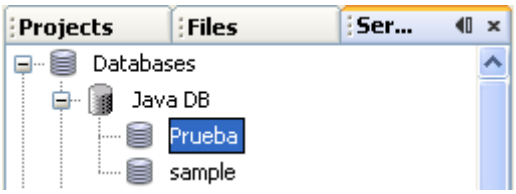


   d. Ahora cree la Base de Datos

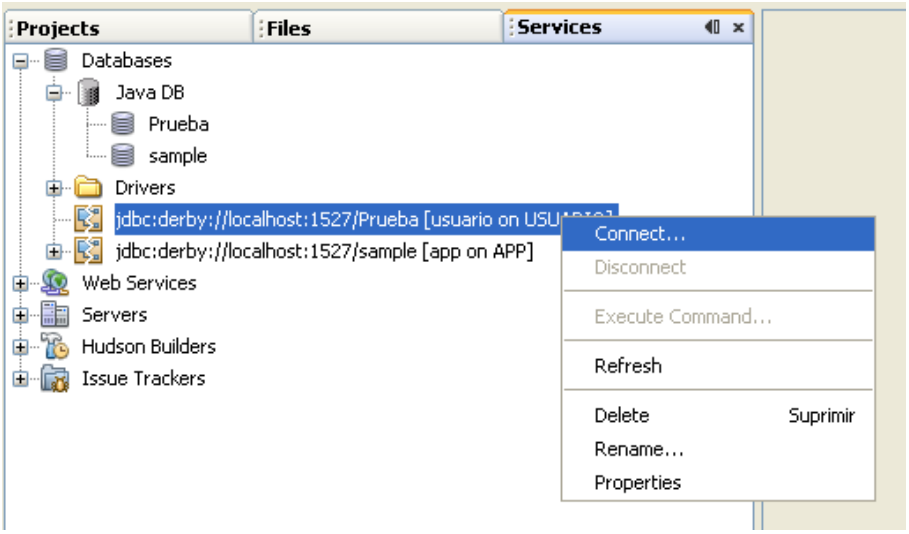
Nombre BD: Prueba
Usuario: usuario
Contraseña: 123456

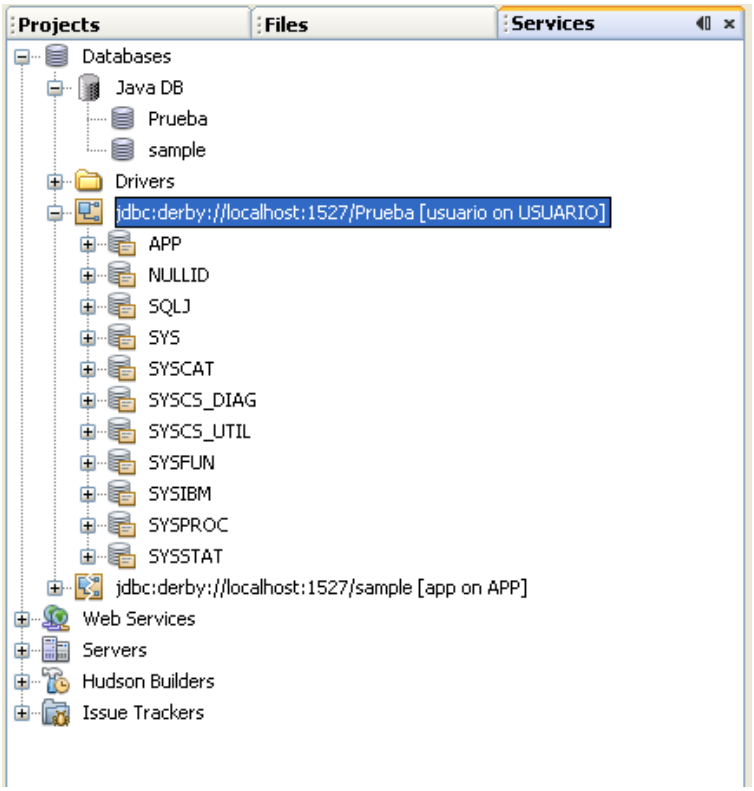




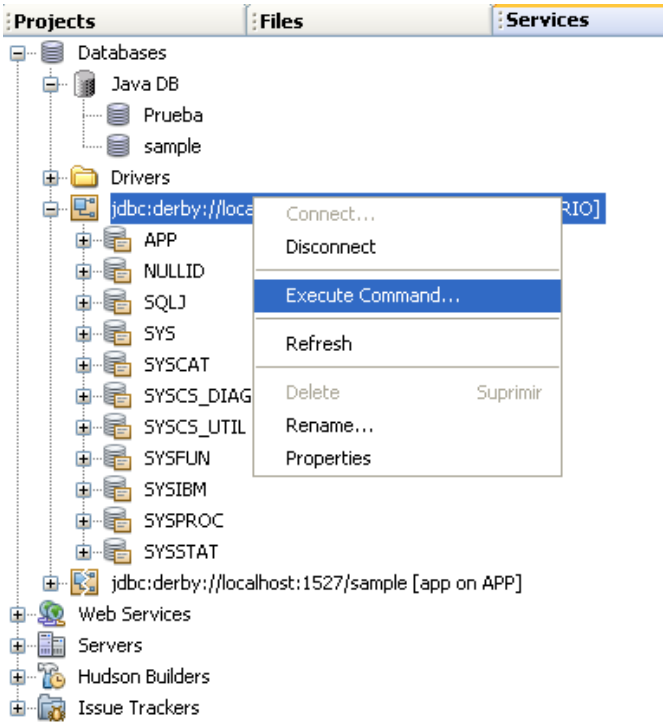


e. Ahora conecte el servicio: Recuerde que debe conectar el servicio que tiene el nombre del usuario que acabo de crear.

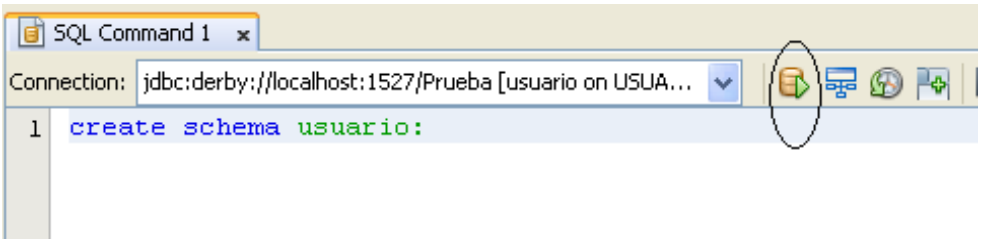


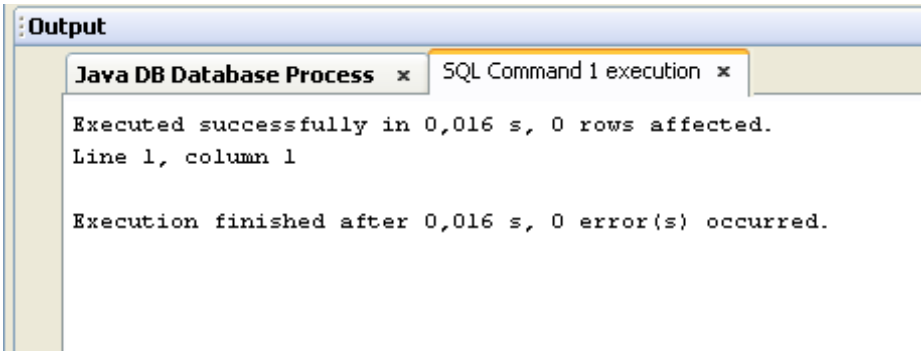


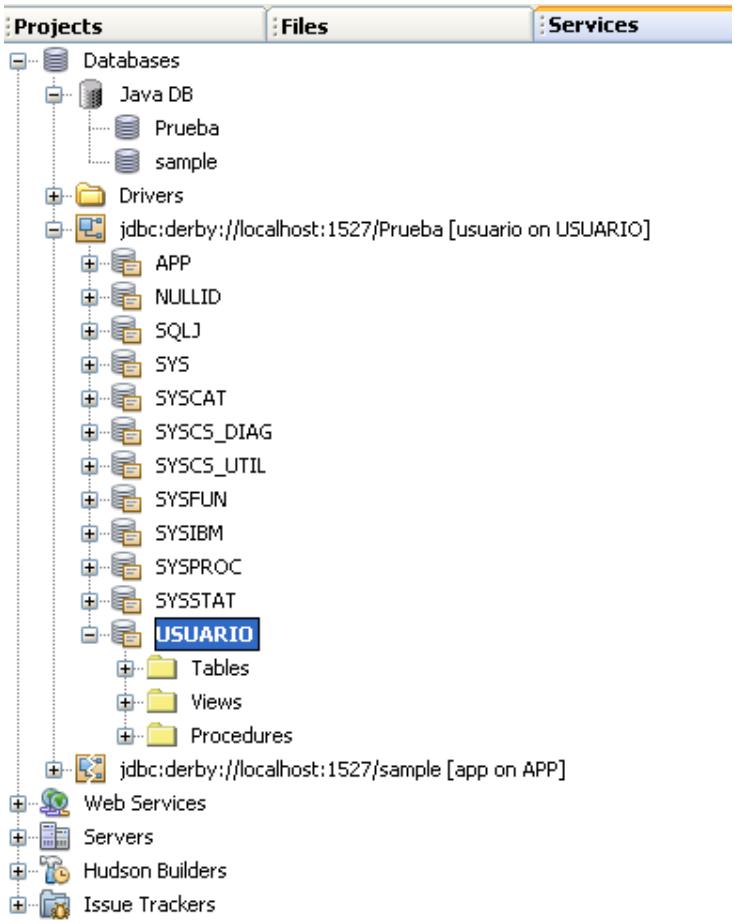
6. A continuación se crea el esquema



UNIVERSIDAD DISTRITAL FRANCSICO JOSE DE CALDAS
FACULTAD DE INGENIERIA - INGENIERIA DE SISTEMAS
PROGRAMACION AVANZADA

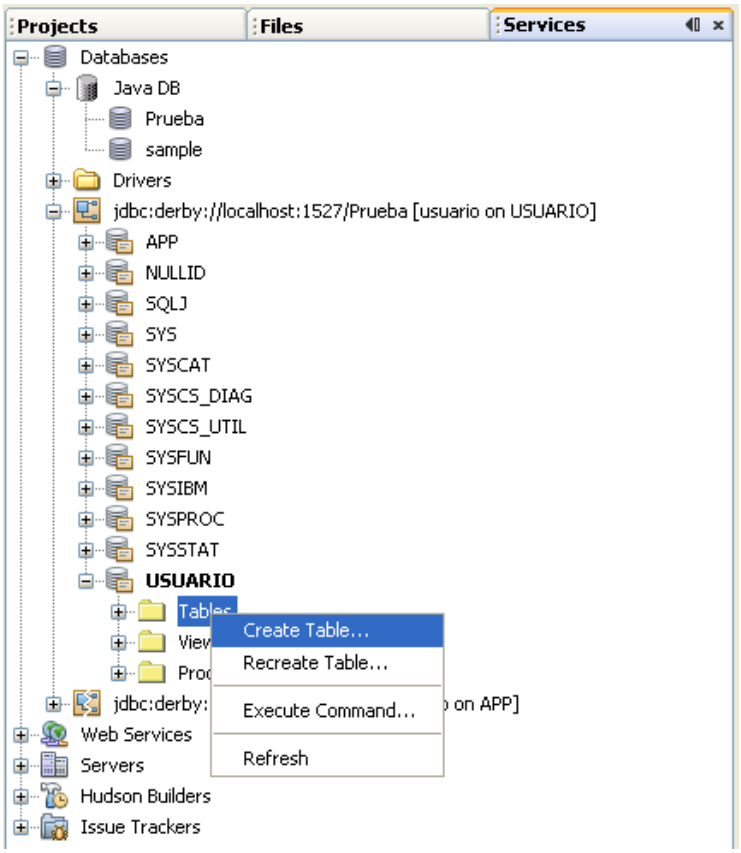


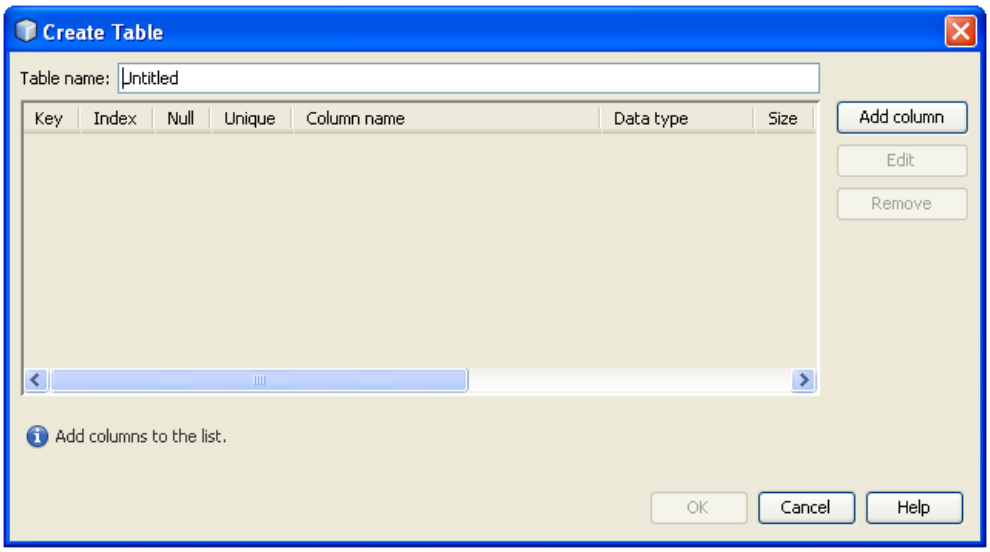




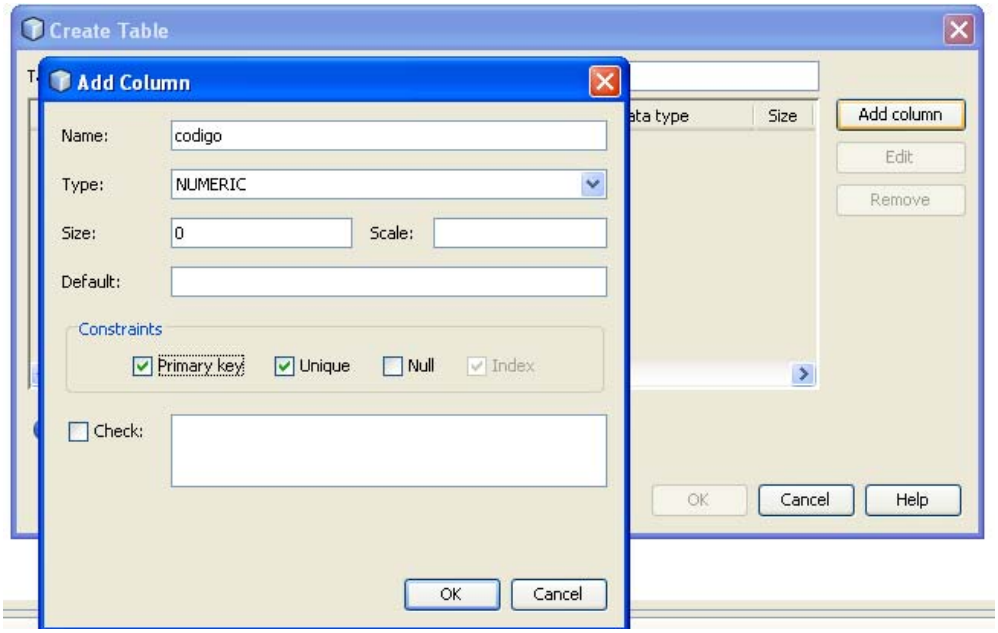
7. Creado el esquema, a continuación cree la primera tabla

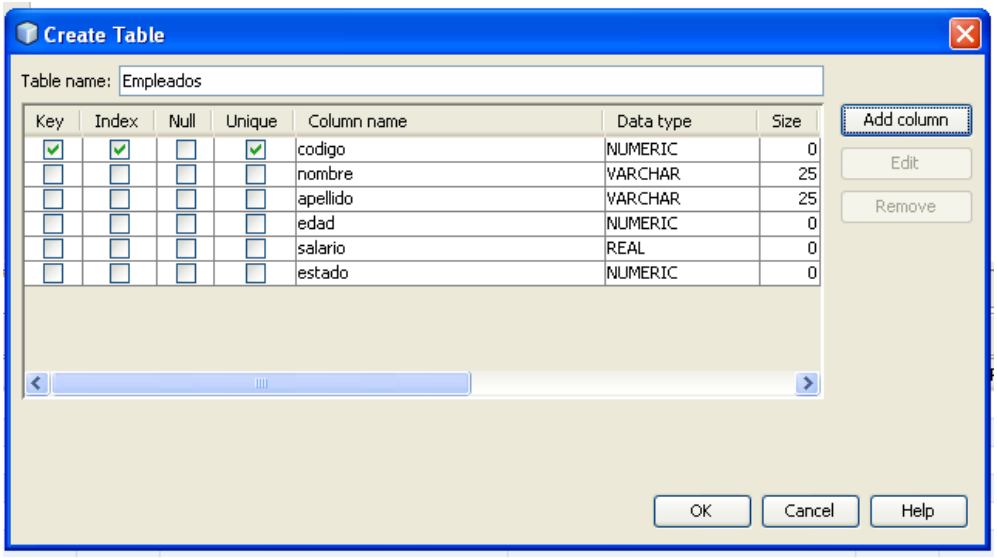
Tabla Empleado
Código: Int
Nombre: Varchar
Apellido: Varchar
Edad: Int
Salario: real
Estado: Int

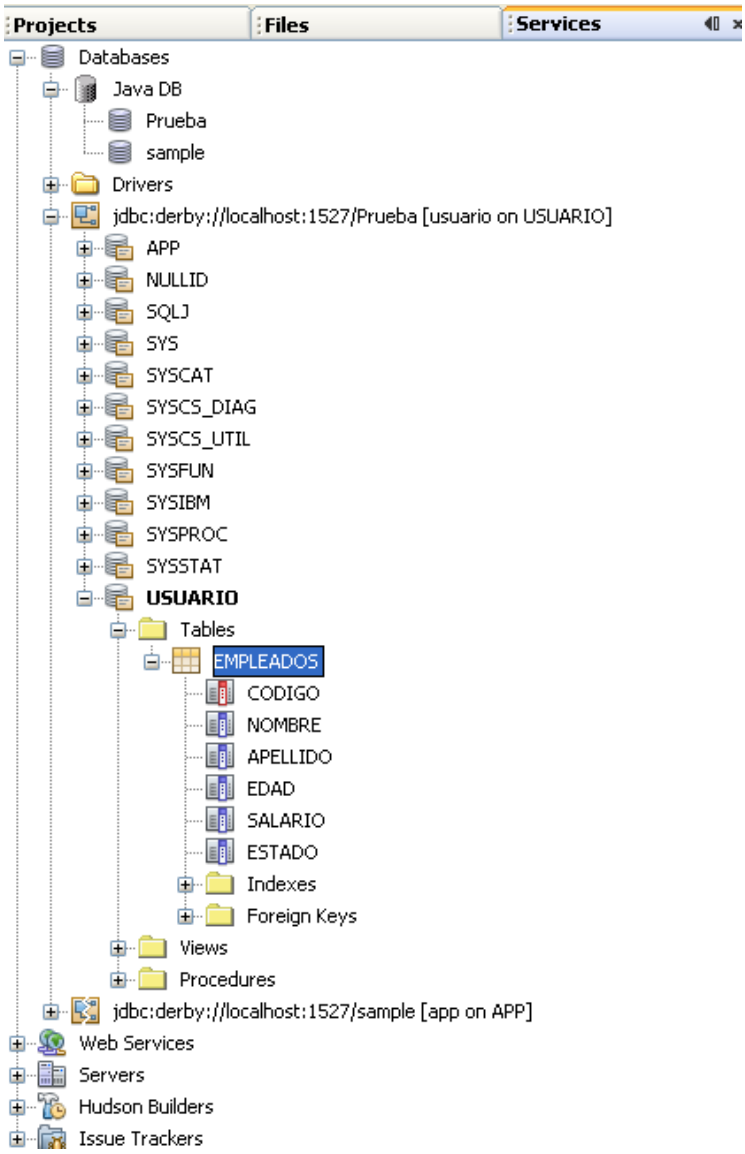




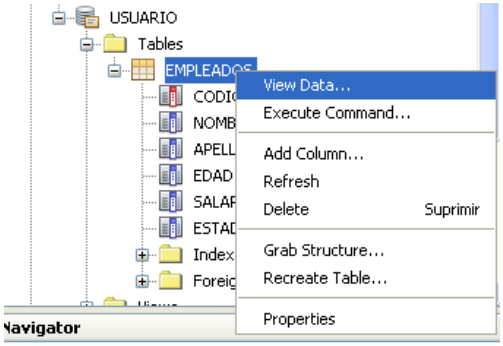
UNIVERSIDAD DISTRITAL FRANCSICO JOSE DE CALDAS
FACULTAD DE INGENIERIA - INGENIERIA DE SISTEMAS
PROGRAMACION AVANZADA



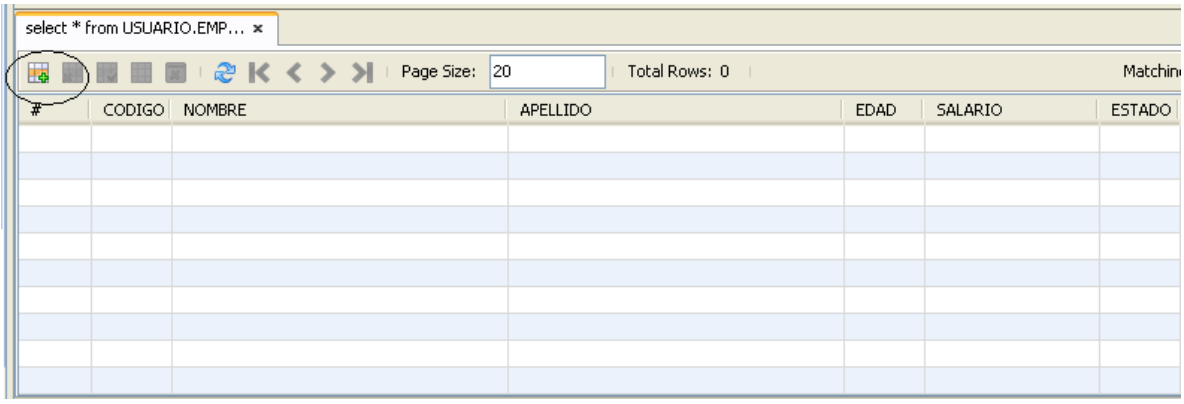


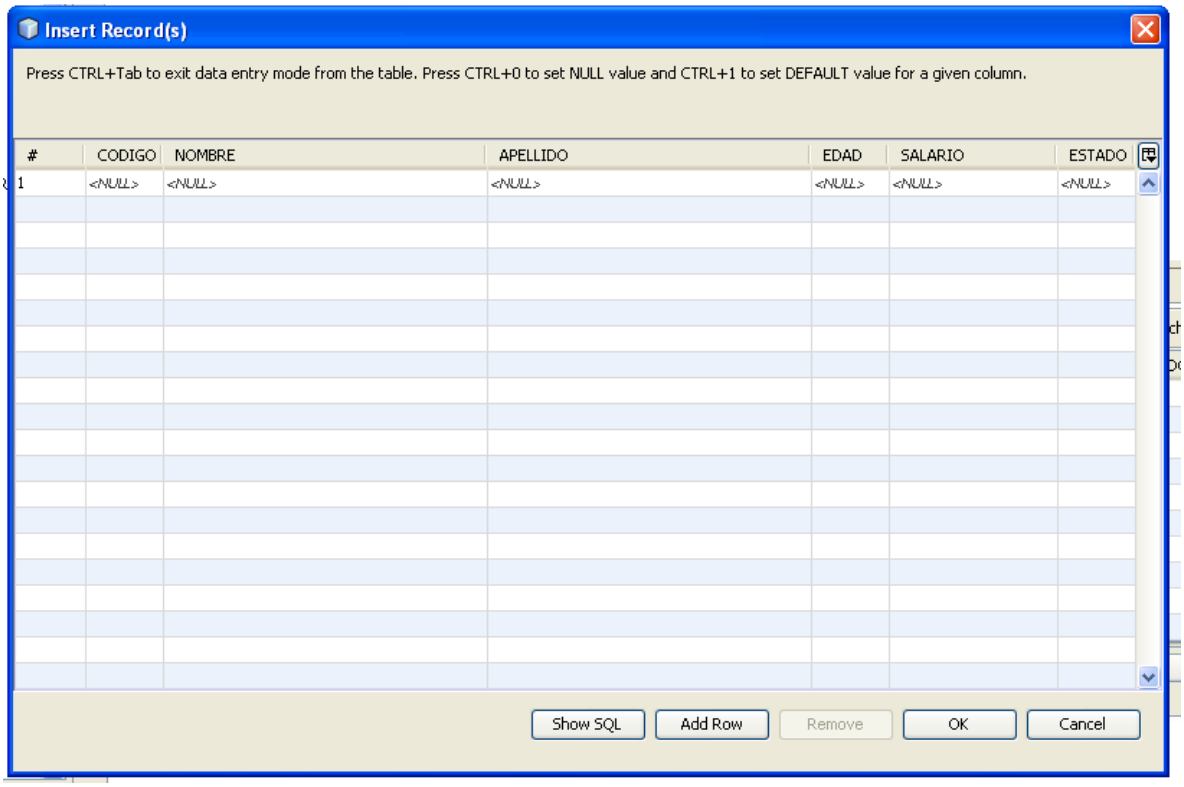


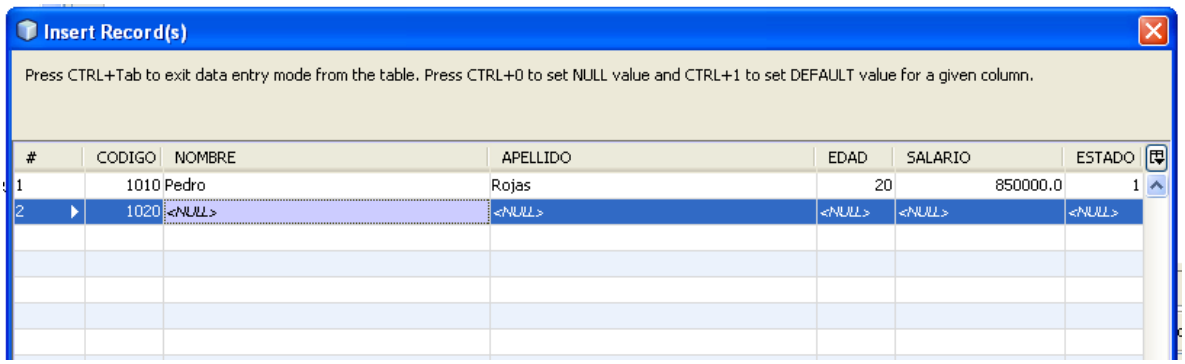
8. A continuación, inserte datos de prueba

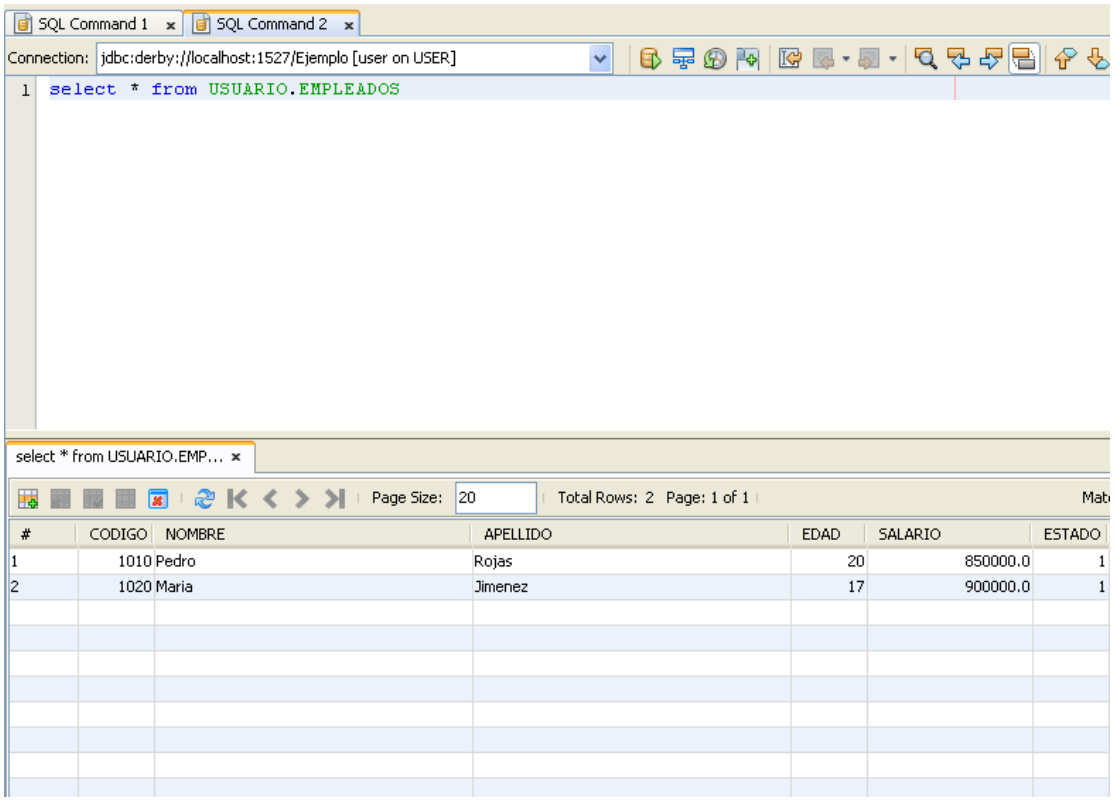


UNIVERSIDAD DISTRITAL FRANCSICO JOSE DE CALDAS
FACULTAD DE INGENIERIA - INGENIERIA DE SISTEMAS
PROGRAMACION AVANZADA

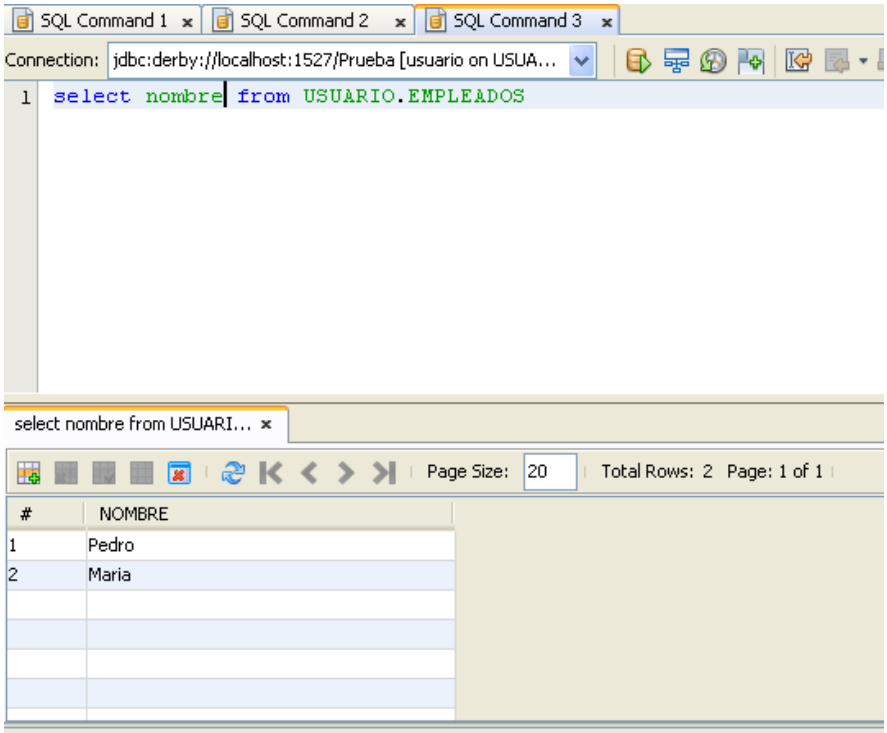








9. Para verificar que los datos fueron ingresados, y que el motor funciona adecuadamente, consulte todos los nombres de los empleados insertados



10. Teniendo los datos insertados, a continuación se deben crear las clases que se conectaran y ejecutaran las transacciones. Para este pasa, se debe retornar a la pestaña de "Proyecto"

11. Cree una clase "Conexion"

12. Inserte el siguiente código al interior de la clase

```
public class Conexion {
   private static Connection cn = null;
   private static Driver driver = new org.apache.derby.jdbc.ClientDriver();
   private static String URLBD = "jdbc:derby://localhost:1527/Prueba";
   private static String usuario = "usuario";
   private static String contrasena = "123456";

   public static Connection getConexion() throws SQLException {
      DriverManager.registerDriver(driver);
      cn = DriverManager.getConnection(URLBD, usuario, contrasena);
      return cn;
   }
}
```

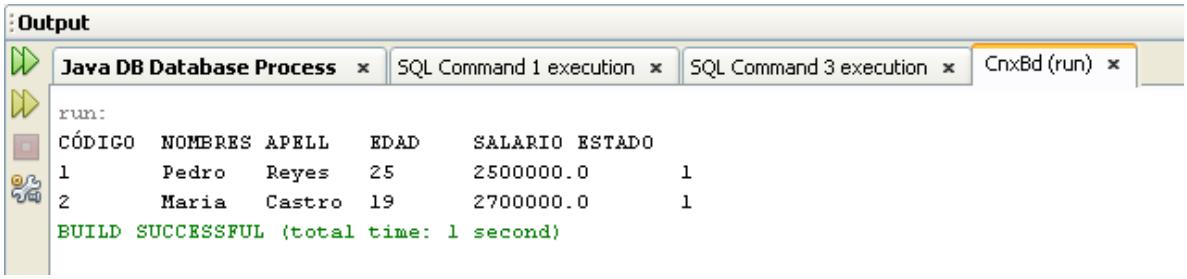
13. A continuación se debe crear la clase que opere la conexión, y administre las operaciones DML
14. Cree una Clase Gestor
15. Inserte el siguiente código

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Gestor {

   private Connection con = null;
   private Statement st = null;
   private ResultSet rs = null;

   public Gestor(){
      try {
```

```
      //Obtenemos la conexion a Derby
      con = Conexion.getConexion();
    } catch (SQLException ex) {
      System.out.println("No se  pudo realizar la conexion");
    }
}

public ResultSet hacerConsulta(){
    String consulta = "SELECT * FROM empleados";
    try {
      //Preparamos la consulta
      st = con.createStatement();
     //Ejecutamos la consulta
     rs = st.executeQuery(consulta);
     mostrarDatos(rs);
    } catch (SQLException ex) {
      System.out.println("No se pudo realizar la consulta");
    }
    return rs;
}

public void mostrarDatos(ResultSet respuesta) throws SQLException{

    System.out.println("CÓDIGO\tNOMBRES\tAPELL\tEDAD\tSALARIO\tESTADO");
      //Ahora recorremos el Resultset
      while (respuesta.next()) {
        int codigo, edad, estado;
        String nombres, apellidos;
        double salario;

        //Recogemos los datos llamando al nombre de la columna que el
        //Resultset nos ha devuelto
        codigo = respuesta.getInt("codigo");
        nombres = respuesta.getString("nombre");
        apellidos = respuesta.getString("apellido");
        edad = respuesta.getInt("edad");
        salario = respuesta.getDouble("salario");
        estado = respuesta.getInt("estado");
        //Imprimimos los datos por la consola
        System.out.println(codigo + "\t" + nombres + "\t" + apellidos + "\t"
```

```
                    + edad + "\t" + salario + "\t" + estado);
            }
        }

        public void cerrarCnx() throws SQLException{
            //Cerramos los elementos usados
              rs.close();
              rs = null;
              st.close();
              st= null;
              con.close();
              con=null;
        }
        public static void main(String[] args) throws SQLException {

             Gestor g=new Gestor();
             g.hacerConsulta();
        }
    }
```

16. A continuación ejecute la clase “Gestor”, y obtendrá como resultado:



17. Cree el método que permita insertar un empleado a la tabla.

Si tiene una instalación que no tenga integrado Derby, a continuación se adjunta una página de referencia donde encontrara información de cómo descargar el jar, instalarlo, crear las clases y realizar la ejecución

http://rchavarria.wordpress.com/2010/06/24/usar-apache-derby-como-base-de-datos/