# Apéndice C: Código Fuente del Programa DBConnection.java

```
import java.sql.*;
import java.io.*;
import java.*;
import java.util.*;
import java.net.*;


public class DBConnection
{

        Connection pgsqlConn = null;        /*Variable para la conexión a Postgres*/
        Connection connection = null;       /*Variable para la conexión a Mysql   */



public void connection()
{
        Statement stmt=null;
        ResultSet rset=null;

        try
        {

         /*Conexión a la base de datos de Postgres*/
        Class.forName("org.postgresql.Driver");
         String pgsqlURL = "jdbc:postgresql://localhost:5432/isystemDB";
         String pgsqlUser = "test";
         String pgsqlPW = "test";
         pgsqlConn = DriverManager.getConnection(pgsqlURL, pgsqlUser, pgsqlPW);

        /*Conexión a la base de datos de Mysql*/
        Class.forName("com.mysql.jdbc.Driver");
        String url="jdbc:mysql://localhost:3306/isystemdb";
        String username = "root";
        String password = "saviaga";
        connection = DriverManager.getConnection(url, username, password);

         /*Obtenemos la información de las tablas de Postgres*/

        stmt = pgsqlConn.createStatement ();
        rset = stmt.executeQuery ("select * from pg_tables");

        /*Mandamos esa información a la función guarda_info(Resultset,Connection)*/
        guarda_info(rset,pgsqlConn);

        /*Cerramos las conexiones*/
       pgsqlConn.close();
        connection.close();

   }//fin del try

   catch (ClassNotFoundException cnfe)
     {
            System.out.println("Could not load database driver:" + cnfe.getMessage());
      }
```

```
   catch (SQLException sqle)
     {
            System.out.println("Could not connect to the database:  " + sqle.getMessage());
     }
}//Fin de connection()


/*****************************************************************************************
 * guarda_info recibe el resultado del query select * from pg_tables                     *
 * dicho query lo recibe en 'rset',  rset contiene una tabla                              *
 * con todos los nombres de las tablas entre otras caracteristicas(schemaname   tablename   tableowner) *
*****************************************************************************************/

public void guarda_info(ResultSet rset,Connection conn)
{

        Statement stmt2 =null;
        int cont=0;
        char x = 'p';
        char y = 'g';
        char z ='_';
        String nombreTabla= "";
        ResultSet rset2 =null;
 try{
        stmt2 = conn.createStatement ();
        while (rset.next ())
        {
            try
              {
                /*Obtiene el nombre de una tabla*/
                nombreTabla =rset.getString(2);
                if (((nombreTabla.charAt(0)==x)||(nombreTabla.charAt(1)==y))
                 &&(nombreTabla.charAt(2)==z))
                {
                  System.out.println("\n Las tablas que no nos interesan son: "+ nombreTabla );
                }
                else
                {
                   /*Hacemos un Select * from  <tabla actual>*/
                   rset2 = stmt2.executeQuery ("select * from " + rset.getString(2));
                  /*La información obtenida del query anterior la mandamos a
                    guarda_col(Resultset,String) junto con el nombre de la tabla*/
                  guarda_col(rset2,rset.getString(2));
                  stmt2.close();
                  cont++;
                }
              }
              catch (Exception e)
              {
                 System.out.println( e.toString() );
              }
        }
      }
      catch (SQLException sqle)
        {
                System.out.println("Could not connect to the database(3): " + sqle.getMessage());
        }
```

```
}//fin de la clase guarda_info(Resultset,Connection)



/*****************************************************************************************
 * Recibe el resultado de query en rset (select * from <tabla actual>                    *
 *junto el nombre de las tablas y dos archivos de texto                                  *
*****************************************************************************************/

public void guarda_col(ResultSet rset, String nombre_tabla)
{
   String columnName,query,type,type2="";
   int numberOfColumns, numberOfColumns2=0;
   ResultSetMetaData metaData,metaData2= null;
   char a = 'p';
   char b = 'g';
   char c ='_';
  try{
      rset.beforeFirst();
      metaData= rset.getMetaData();
      numberOfColumns = metaData.getColumnCount();
       /*Ahora comenzamos a construir el query que utilizaremos para
         introducir los datos en la tabla de Mysql                                    */
      query = "insert into " + nombre_tabla + "(";
      for ( int i=1;i<=numberOfColumns;i++)
            {
              columnName = metaData.getColumnLabel(i);
              type = metaData.getColumnTypeName(i);
              query+= columnName + ",";
            }
      query=query.substring(0,query.length()-1)+ ") values (";
      for ( int x=1; x<=numberOfColumns;x++)
       {
        query+="?,";
       }
       query=query.substring(0,query.length()-1)+ ")";
       if (((nombre_tabla.charAt(0)==a)||(nombre_tabla.charAt(1)==b))&&(nombre_tabla.charAt(2)==c))
       {
            System.out.println("\n Estas tablas no nos interesan"+ nombre_tabla);
       }
       else
            {
              PreparedStatement pstmt = connection.prepareStatement(query);
               while (rset.next())
               {
               try
                {
                 metaData2= rset.getMetaData();
                 numberOfColumns2 = metaData2.getColumnCount();
                for ( int z=1;z<=numberOfColumns2;z++)
                {
                     type2 = metaData2.getColumnTypeName(z);
                     /*Checa que tipo de dato es para meterlo en el query*/
                    If  ((type2.compareTo("name")==0)||(type2.compareTo("varchar")==0)||
                       (type2.compareTo("char")==0))
                       {pstmt.setString(z,rset.getString(z));}
                    else
```

```
                {
                  if ((type2.compareTo("int4")==0)||(type2.compareTo("int8")==0)||
                      (type2.compareTo("numeric")==0))
                  {pstmt.setLong(z,rset.getLong(z));}
                 else
                   {
                  if (type2.compareTo("bool")==0)
                    {pstmt.setBoolean(z,rset.getBoolean(z));}
                  else
                    {
                     if ((type2.compareTo("text")==0)||(type2.compareTo("_text")==0)||
                         (type2.compareTo("anyarray")==0))
                         {pstmt.setString(z,rset.getString(z));}
                      else
                         {
                           if (type2.compareTo("abstime")==0)
                               {pstmt.setTime(z,rset.getTime(z));}
                           else
                               {
                               if ((type2.compareTo("float4")==0)||(type2.compareTo("_float4")==0)||
                                  (type2.compareTo("float8")==0))
                                  {pstmt.setFloat(z,rset.getFloat(z));}
                               else
                                  {
                                   if ((type2.compareTo("oid")==0)/*||(type2.compareTo("unknown")*/)
                                       {pgsqlConn.setAutoCommit(false);
                                        pstmt.setBlob(z,rset.getBlob(z));}
                                   else
                                       {
                                        if (type2.compareTo("double")==0)
                                            {pstmt.setDouble(z,rset.getDouble(z));}
                                            else
                                                {pstmt.setBlob(z,rset.getBlob(z));}
                                       }
                                  }
                               }
                         }
                    }
                  }
                }

       } //Fin del for
       pstmt.execute();
       pstmt.clearParameters();
      pgsqlConn.commit() ;
   } //Fin del try
    catch (SQLException sqle)
     {
        System.out.println("Could not connect to the database(4) : " + sqle.getMessage());
     try{
         String entra;
          BufferedReader input = new BufferedReader( new InputStreamReader(System.in));
          entra =  input.readLine();
        }
       catch (IOException e)
          {
             System.out.println("Could not read: " + e.getMessage());
          }
     }
}//Fin del While
```

```
      }
  }//fin del try


catch (SQLException sqle)
{
   System.out.println("Could not connect to the database (5): " + sqle.getMessage());
  try{
            String entra;
            BufferedReader input = new BufferedReader( new InputStreamReader(System.in));
            entra =  input.readLine();
     }
     catch (IOException e)
     {
            System.out.println("Could not read: " + e.getMessage());
     }
}
catch (Exception ex)
  {
    System.out.println("Error: "+ ex.toString());
    try{
            String entra;
            BufferedReader input = new BufferedReader( new InputStreamReader(System.in));
            entra =  input.readLine();
         }
        catch (IOException e)
            {
               System.out.println("Could not read: " + e.getMessage());
            }
   }
} //fin de la clase guarda_col()

public static void main(String args[])
{
            new DBConnection().connection();
}

}//class
```