I.OBJETIVOS

- Que el alumno sea capaz de conocer las clases específicas que posee Java para el manejo de base de datos y el manejo de las API de JDBC.
- Que al alumno sea capaz de aprender el uso de las API JDBC para la creación de consultas.

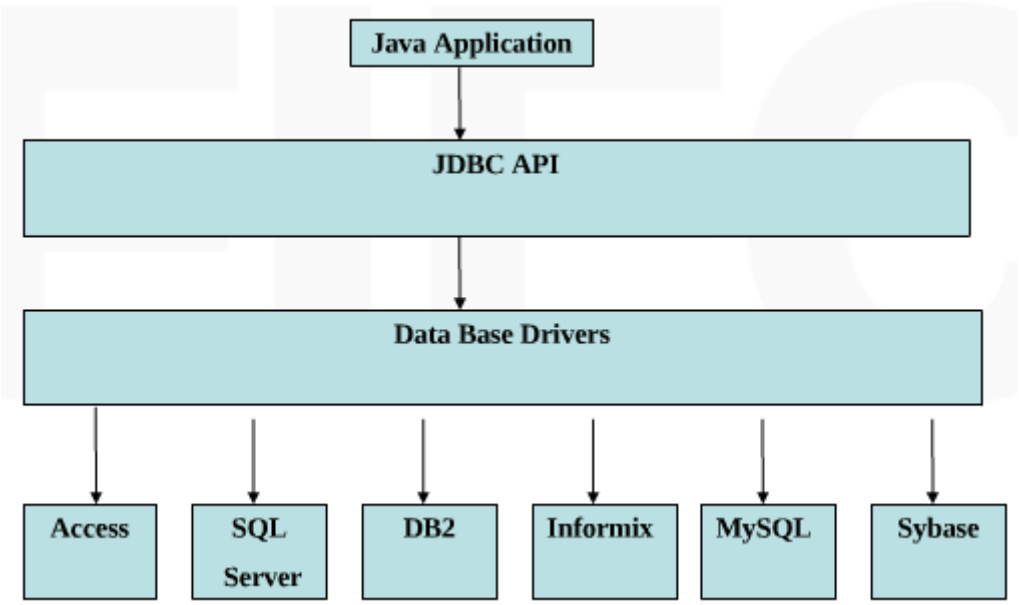
II. INTRODUCCIÓN

Conexión a Bases de Datos con NetBeans IDE 8 o superior

JDBC (Java DataBase Connectivity) es la tecnología Java que permite a las aplicaciones interactuar directamente con motores de base de datos relacionales.

La API JDBC es una parte integral de la plataforma Java, por lo tanto no es necesario descargar ningún paquete adicional para usarla. JDBC es una interface única que independiza a las aplicaciones del motor de la base de datos.

Un driver JDBC es usado por la JVM para introducir las invocaciones JDBC en invocaciones que la base de datos entiende.



Conectar a la base de datos

Para que una aplicación en Java se comunique con una base de datos usando la API JDBC, se requiere de un conector que realice la conexión de la aplicación a la Base de Datos.

Ese conector es específico para el manejador de base de datos y viene en la forma de un archivo ".jar" o ".zip".

Por ejemplo:

MySQL, está en el archivo: mysql-connector-java5.1.6.zip (mysql-connector-java-5.1.6-bin.jar)

SQL Server, está en el archivo: sqljdbc_1.2.2828.100_esn.zip (sqljdbc.jar)

NetBeans nos permite realizar algunas tareas relacionadas con bases de datos:

1. Conectar una aplicación a una base de datos.
2. Conectar a NetBeans directamente a una base de datos para crear, eliminar, modificar tablas, agregar, eliminar, modificar registros y hacer consultas de datos.
3. Generar aplicaciones a partir de ingeniería inversa.
4. Crear persistencia con la Base de Datos.
5. Otras innumerables.

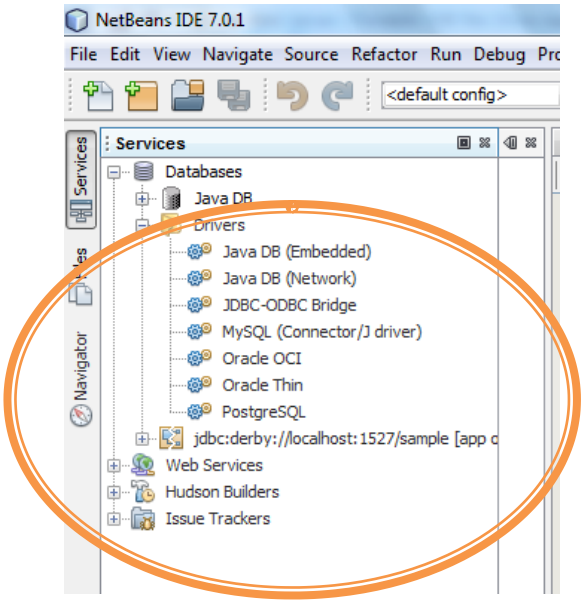
Conexión de una aplicación a una base de datos

Para conectar a una aplicación a una base de datos, se requiere:

1. Agregar a NetBeans el conector como una biblioteca o librería. Esto permite que el conector esté disponible para los proyectos.
2. Agregar a un proyecto el conector. Esto permite que la aplicación se pueda conectar a la base de datos.

Agregar a NetBeans el conector para utilizar una base de datos en MySQL.

En las versiones más recientes de NetBeans, se cuenta ya con las librerías para la mayoría de gestores de bases de datos, o al menos, los más importantes.



Esto no es una regla exclusiva, puede tener una versión un poco antigua de Netbeans (por lo que se le recomienda actualizar), pueden no funcionar los drivers que posee o simplemente no estar incluidos. Certifique que lo posee, revise en su grupo de ventanas en **Services->Drivers** que posee al menos el conector MySQL (Para esta práctica)

¿No se encuentras las librerías?

Nota: Siga este procedimiento solo si no se encuentran o no funciona la librería de MySQL.

Agregaremos a Netbeans el conector de MySQL como ejemplo. El procedimiento es el siguiente:

1. Seleccionar la opción **Tools/Libraries** de la barra de menú de **NetBeans IDE 8.0 o superior** como se muestra en la siguiente figura:

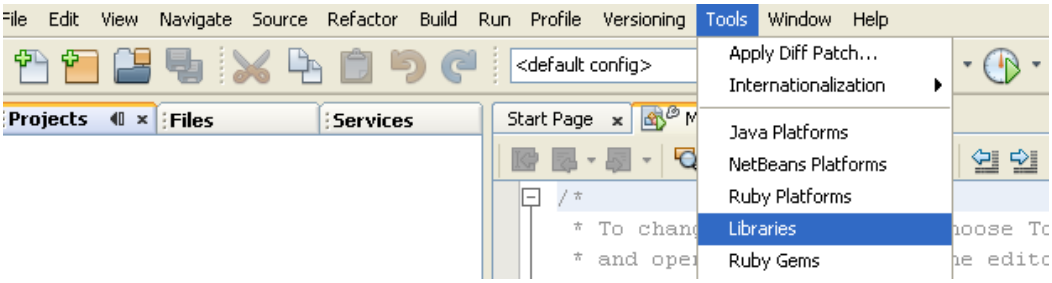


Figura 1

2. Aparece un cuadro de diálogo que nos permite administrar las bibliotecas de **NetBeans IDE 8.0.**

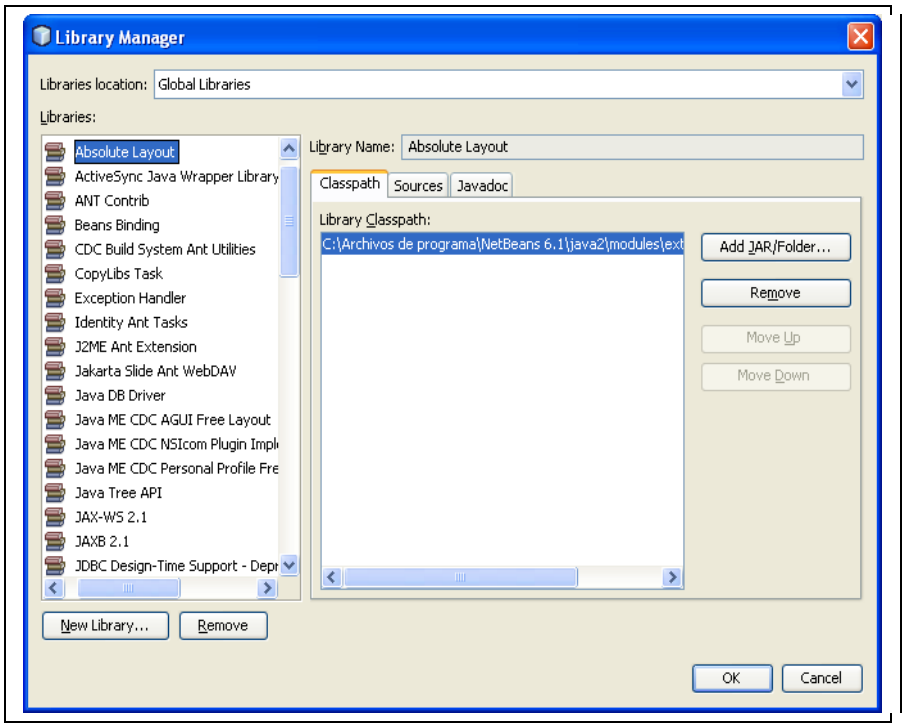


Figura 2

Del lado izquierdo del cuadro de dialogo, aparece una ventana con las bibliotecas agregadas a NetBeans. Del lado derecho aparece el nombre de la biblioteca y la trayectoria del archivo con la biblioteca. Para agregar el conector de **MySQL** a NetBeans, hacer clic en el botón **New Library…** y aparece un cuadro de dialogo como el siguiente:

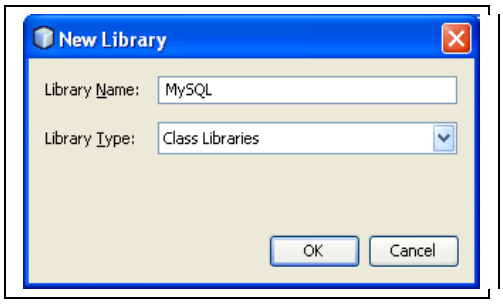


Figura 3

3. En este cuadro de dialogo se establece el nombre que tendrá el conector, digitar **MySQL** y hacer clic en el botón **OK.** Y se regresara a la ventana del administrador de bibliotecas. Hacer clic en el botón **Add JAR/Fólder…**

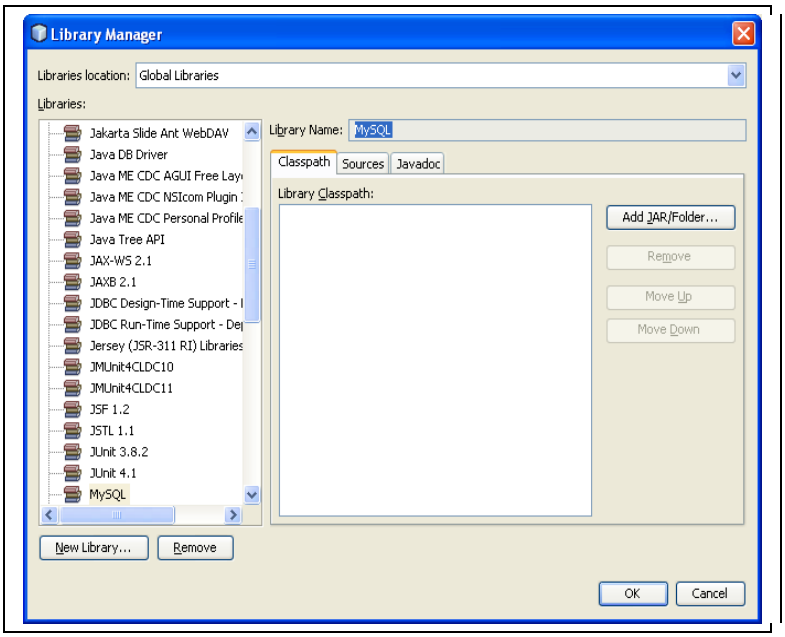


Figura 4

4. Hay que seleccionar el archivo que contiene el conector de MySQL: mysql-connector-java-5.0.8-bin.jar (Puede descargalo en el sitio de oracle), después de haberlo seleccionado, hacer clic en **Add JAR/Fólder.**

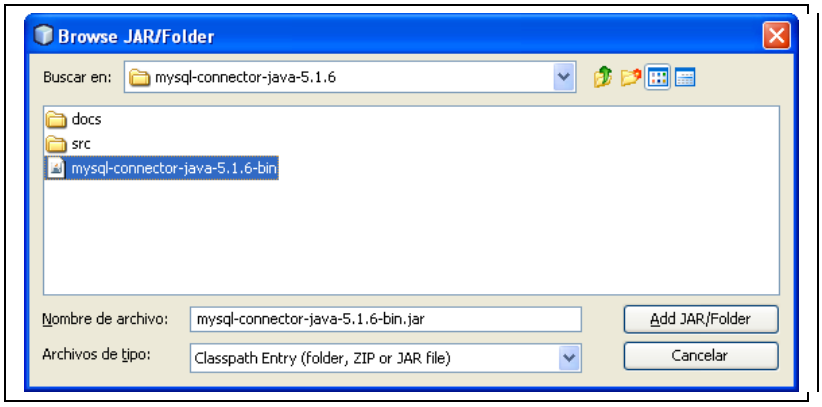


Figura 5

5. Regresamos al administrador de bibliotecas en donde aparece el conector agregado, hacer clic en el botón **OK.**

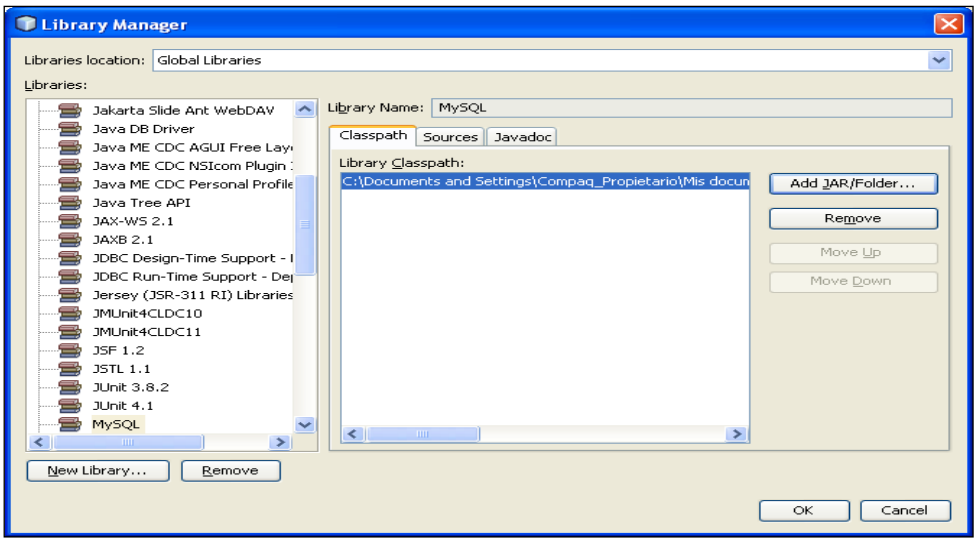


Figura 6

Nota:
Los pasos anteriores debe seguirse para hacer la conexión de una aplicación a una Base de Datos de SQL Server también tomar en cuenta los siguientes pasos:

- Seleccionar la opción **Tools/ Libraries** de la barra de menú como se muestra en la figura 1.
- Mostrará un cuadro de dialogo que permite administrar las bibliotecas de **NetBeans IDE 8** como se muestra en la figura 2.
- Agregar una nueva librería (**New Library**), la cual se llamara **MSSqlServer**, el nombre lo colocara en la opción **Library name** así como se muestra en la figura 3.
- Seleccionar el archivo que contiene el conector a SQL Server:

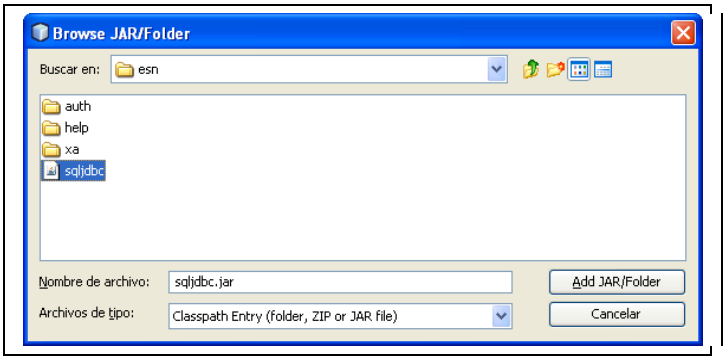


Figura 7

- Regresar al administrador de bibliotecas en donde aparece el conector agregado, así como se muestra en la figura 6, y hacer clic en el botón **OK**

Importante:

Para hacer uso de las conexiones de las bases de datos que se instalaron anteriormente, debe copiar los archivos .jar en la siguiente dirección:
C:\Archivos de programa\Java\jre1.6.0_05\lib\ext
Ó
C:\Program Files\Java\jdk1.6.0_25\jre\lib\ext

Así como se muestra en la siguiente figura:

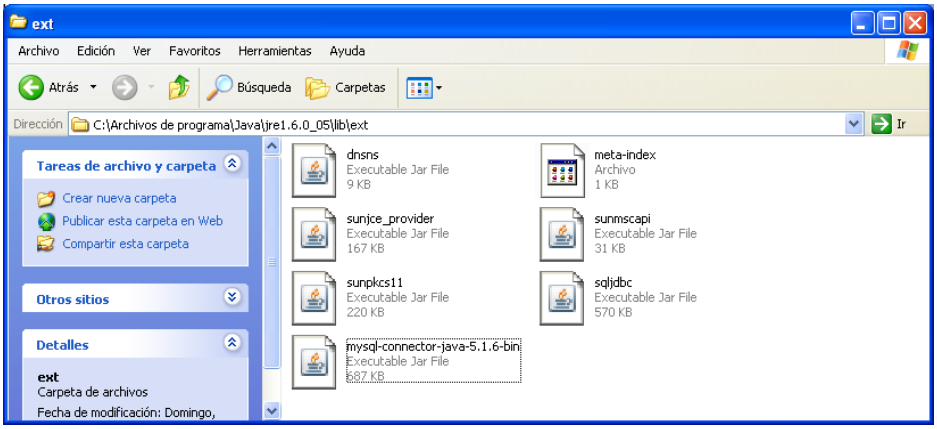


Figura 8

III. PROCEDIMIENTO

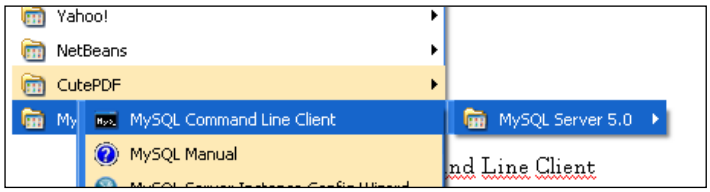
En esta guía trabajaremos la conexión a **MySQL** desde una aplicación de Java creada en NetBeans 8 Ejercicio 1. Crear un proyecto en Netbeans con el nombre Guia4 y realizar todos los pasos que están descritos en la introducción de esta guía.

Ejercicio 1. Creando una Base de datos en MySQL 5.0

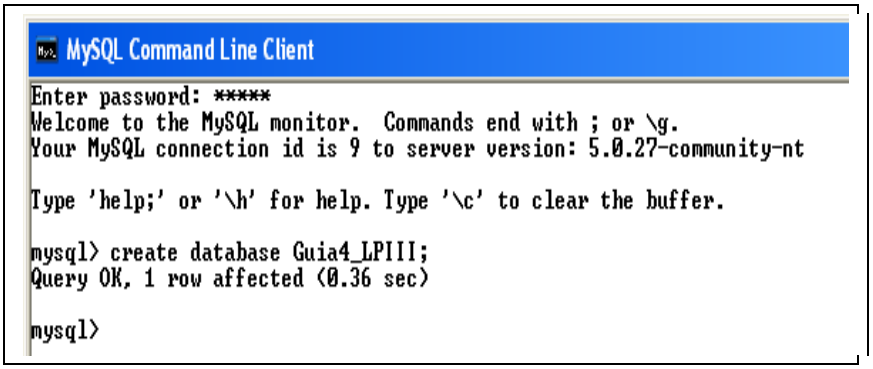
¿Tiene instalado en su Equipo MySQL Server?

Realizar los siguientes pasos:

1. Hacer clic en el botón Inicio, seleccionar Todos los programas, después seleccionar la opción MySQL, seleccionar la opción MySQL 5.0 y por ultimo dar clic en MySQLCommand Line Client



2. Aparece una ventana en la cual debe digitar la contraseña del usuario **root** de MySQL, en la opción **Enterpassword** digitar **admin (**o el password asignado, pregunte a su instructor**)**, presionar la tecla Enter.
3. Digitar **create database Guia4;**(Toda línea termina con punto y coma), presionar la tecla Enter



En este paso hemos creado una Base de datos llamada Guia4, para hacer uso de la base de datos debe digitar: **use Guia4;**

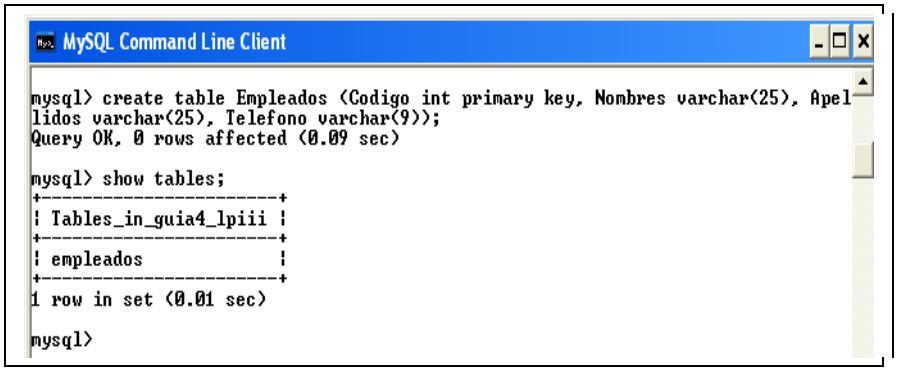
4. En este paso vamos a crear una tabla en la BD creada en el paso 4, vamos a crear Tabla Empleados con la siguiente estructura:

Nombre_columna	Tipo_dato
Codigo	int (llave primaria)
Nombres	varchar(25)

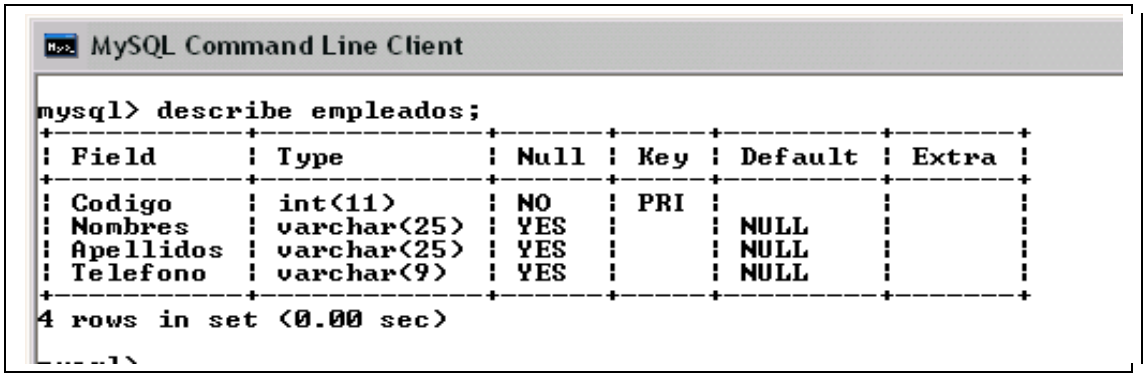
Apellidos	varchar(25)
Telefono	varchar(9)

```
create table Empleados
(Codigo int primary key,
Nombre varchar(25),
 Apellidos varchar(25),
Telefonovarchar(9)
)
```

Nota: Para ver las tablas creadas en la base de datos digitar: **show tables;**



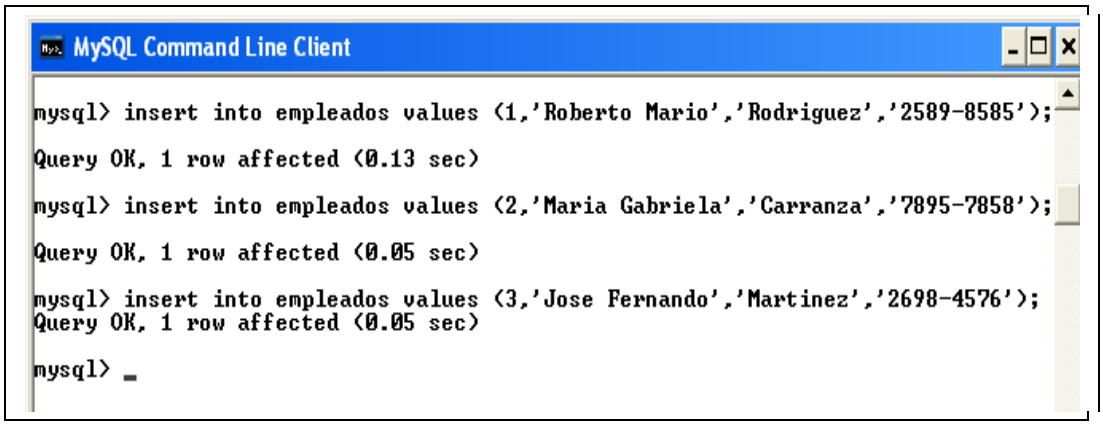
5. Para ver la estructura de la tabla debe digitar: **describe empleados;**



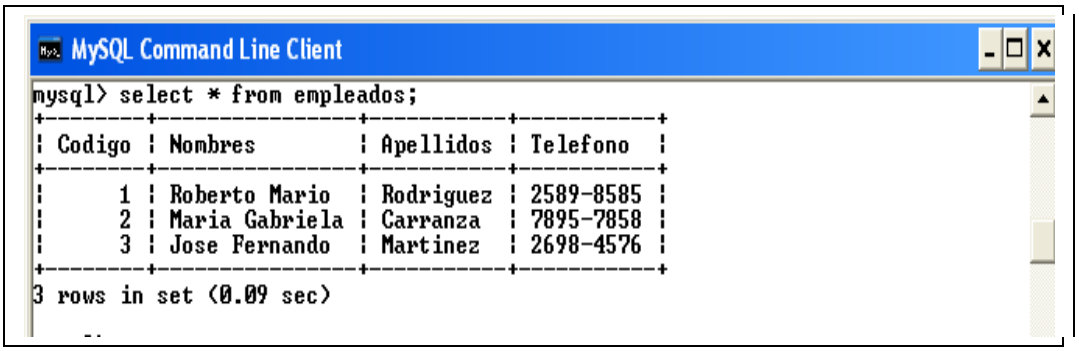
6. Agregar los siguientes datos:

1	Roberto Mario	Rodríguez	2589-8585
2	Maria Gabriela	Carranza	7895-7858
3	José Fernando	Martínez	2698-4576

Para agregar los datos debemos utilizar transacciones SQL, en este caso se debe usar la cláusula **insert**, así como se muestra en la siguientefigura:

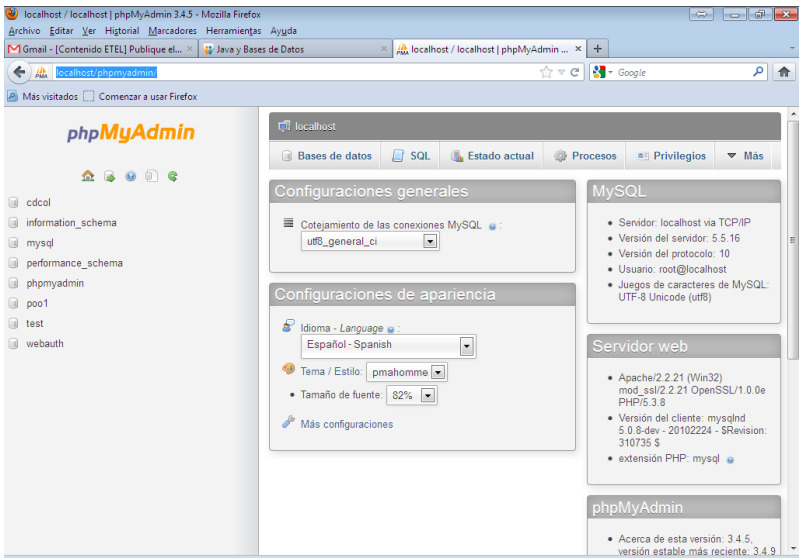


7. Para seleccionar los datos de la tabla debe realizar una consulta **select**. En este caso se quiere seleccionar toda la información de la tabla, se debe digitar: **select * from empleados;**



¿Tiene otro MySQLi nstalado en su equipo en otro servicio como XAMP?

XAMP es un aplicativo que incluye otros servicios como Apache,MySQL y PHP. La administración de MySQL puede realizarse desde una interfase de aplicación PHP incluida (http://localhost/phpmyadmin/)



Luego continuar desde el paso 3 de este ejercicio, debe examinar las opciones de la aplicación para lograr crear la base de datos y las tablas.

¿Se conectará a un servidor MySQL instalado en otro equipo?

Debe conocer la IP y el puerto del servidor MySQL.Este será el caso más habitual en un entorno de trabajo real. Consulte a su instructor para más detalles.

Luego continuar desde el paso 3 de este ejercicio, utilizará la consola de comandos SQL de NetBeans para este caso.

Ejercicio 2. Conexión de NetBeans IDE a una base de datos

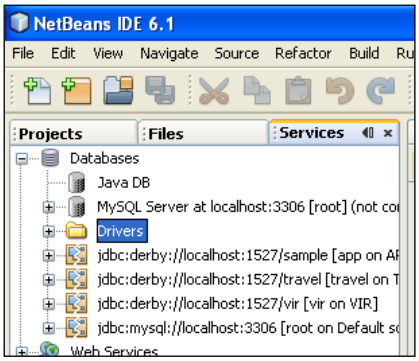
Para conectar una aplicación a una base de datos, se requiere:

- Instalar en NetBeans IDE el conector de la base de datos **SOLO SI NO LO TIENE INSTALADO**
- Establecer la conexión entre NetBeans IDE y la base de datos

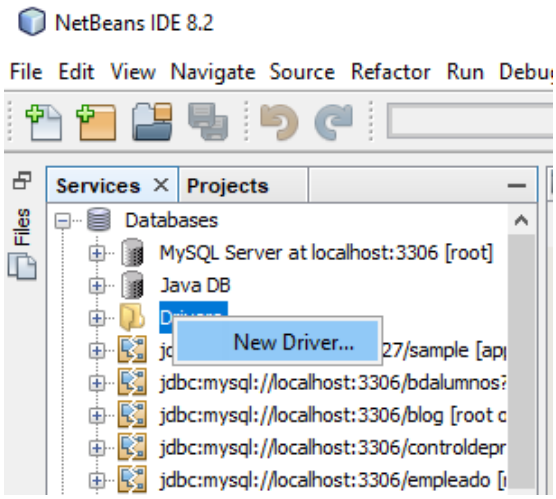
Instalación en NetBeans IDE del conector de la Base de Datos (MySQL)

El procedimiento a seguir es el siguiente:

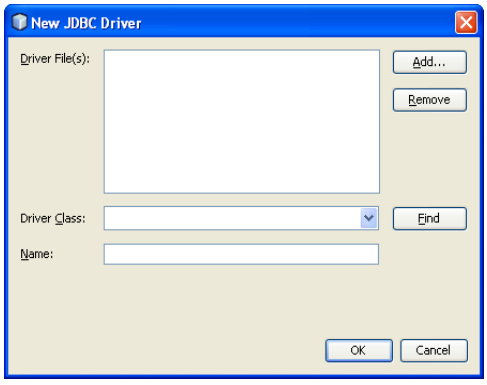
1. Hacer clic en la opción **Services** del proyecto y expanda los nodos **Databases** y **Drivers** para los conectores disponibles y las conexiones de las Base de Datos.



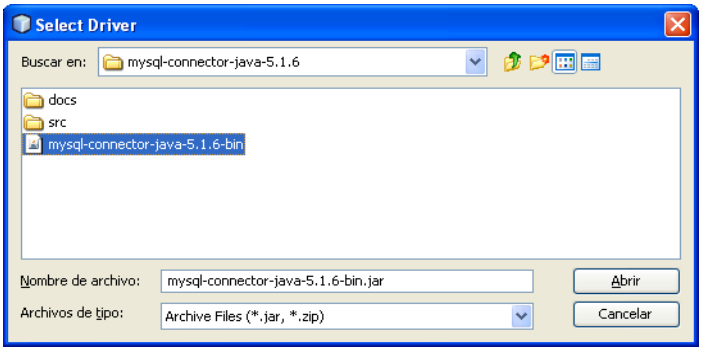
2. Para agregar el conector de MySQL, hacer clic derecho en el nodo **Drivers** y seleccione la opción **New Driver** del menú contextual.



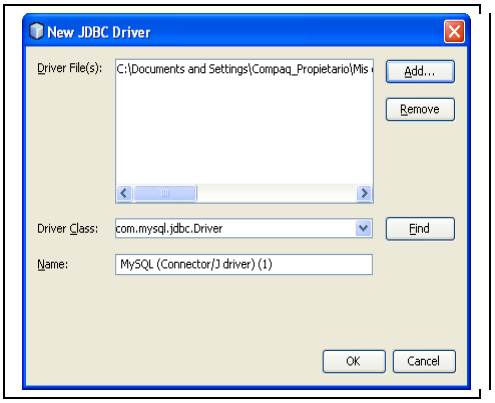
3. Aparece un cuadro de dialogo para agregar un conector, hacer en el botón **Add**



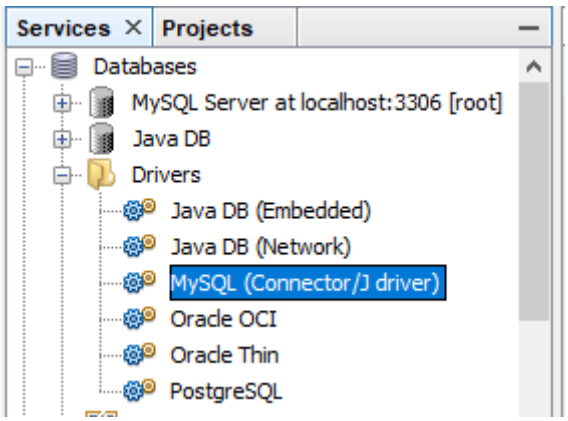
4. Seleccionar el conector, seleccionar el conector y hacer clic en el botón Abrir.



5. Aparece la información del conector seleccionado. Para confirmar hacer clic en el botón **OK.**

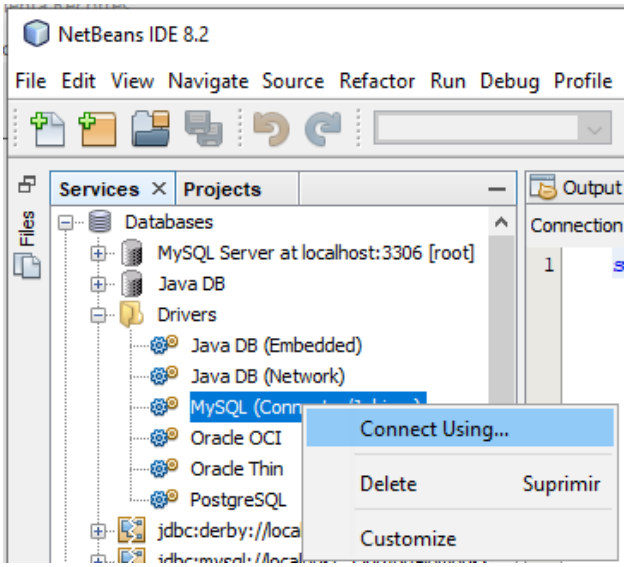


6. Se agrega un nuevo nodo para el conector de **MySQL** en la opción **Services**



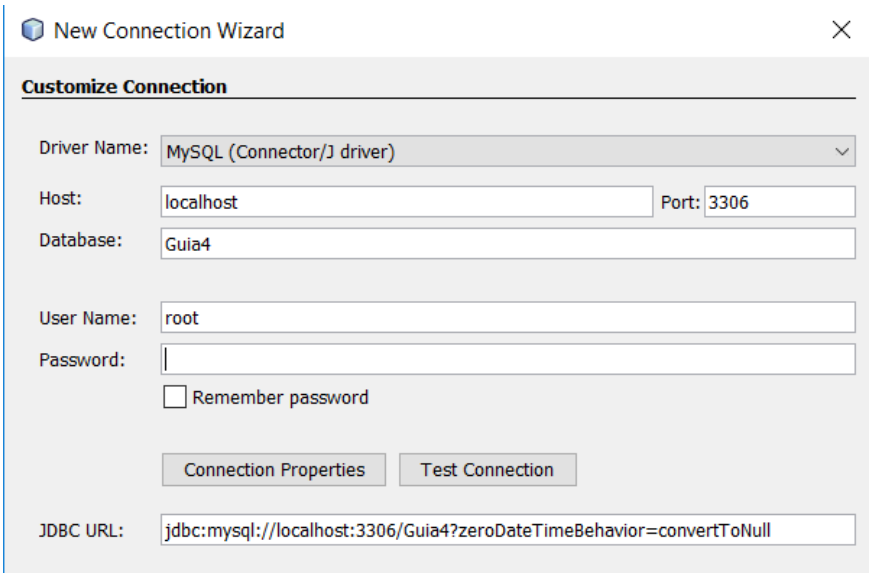
Establecer la conexión entre NetBeans IDE y la base de datos (ESTOS PASOS SÍ LOS REALIZARÁ)

1. Hacer clic derecho sobre el nodo del conector a **MySQL**

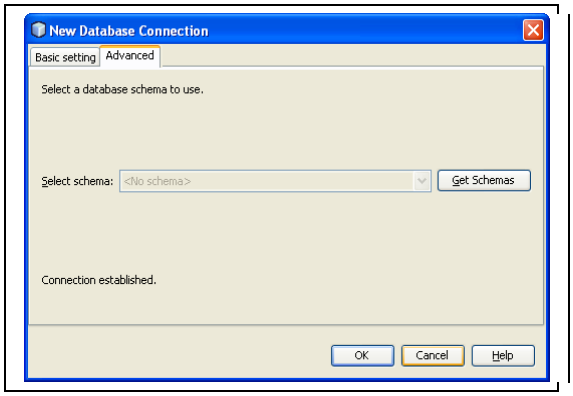


2. Seleccionar la opción **ConnectUsing…**, del menú contextual, aparece un cuadro de dialogo para establecer una nueva conexión, aparece una ventana como se muestra a continuación:
   Los datos que deben estar en esta ventana son:
   a. En la opción Name: **MySQL (Connector/J driver)**
   b. En la opción Driver: **com.mysql.jdbc.Driver**
   c. Establecer en la opcionDatabase URL el siguiente formato:
      jdbc:mysql://servidor:puerto/baseDatos
      Por ejemplo:
      jdbc:mysql://localhost:3306/mysql

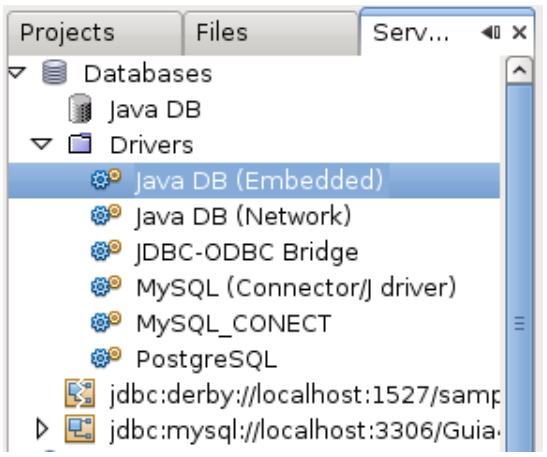
   Donde **servidor** es la dirección IP (o nombre de dominio del servidor), en caso que el servidor este en la misma computadora que NetBeans utiliza el nombre localhost, **puerto**, es el puerto 3306 del servidor, si el servidor utiliza el puerto predefinido se puede omitir, **baseDatos**, es la base de datos a la que se desea conectar. Establecer el nombre del usuario (Username) y la contraseña (Password), para acceder a la base de datos, presionar el botón **OK.**



3. Aparece un cuadro de dialogo confirmando que se estableció la conexión. Para aceptar hacer clic en el botón **OK.**



4. Para confirmar los datos, aparece un nuevo nodo con la conexión a la base de datos.

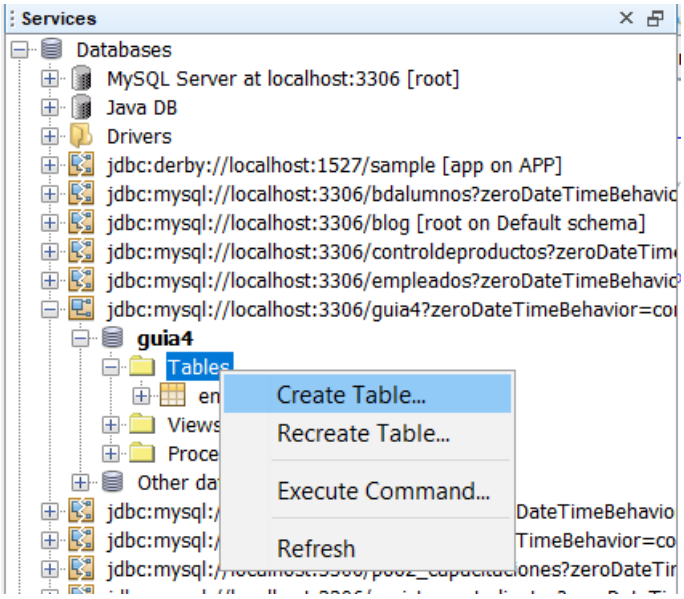


Acceso a la Base de Datos desde NetBeans IDE

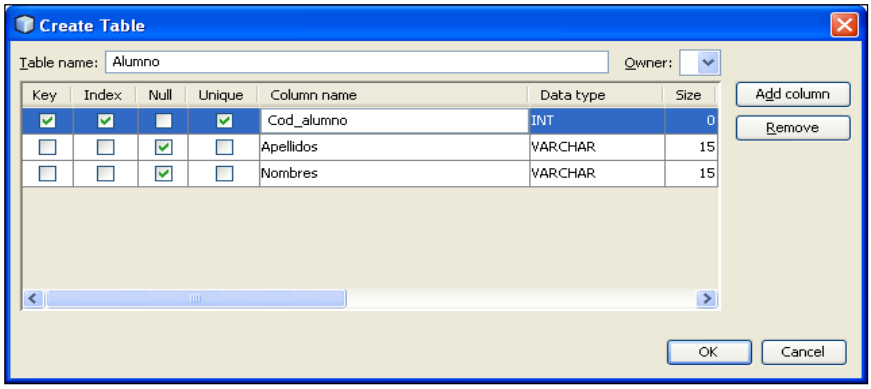
NetBeans nos permite realizar operaciones sobre la base de datos como crear y borrar tablas, agregar, eliminar, actualizar y seleccionar registros que están almacenados en las tablas, para hacer estas operaciones, debe expandir el nodo con la conexión a la base de datos para que aparezcan los objetos creados en la BD (tablas, vistas y procedimientos).

Para ver las tablas creadas en una BD, hacer clic sobre el nodo de la opción **Tables** y para ver las columnas o campos de una tabla hacer clic en (+) en el nombre de cada tabla.

Si quiere crear una tabla nueva en la BD, haga clic derecho sobre el nombre **Table** y seleccionar la opción **CreateTable…**

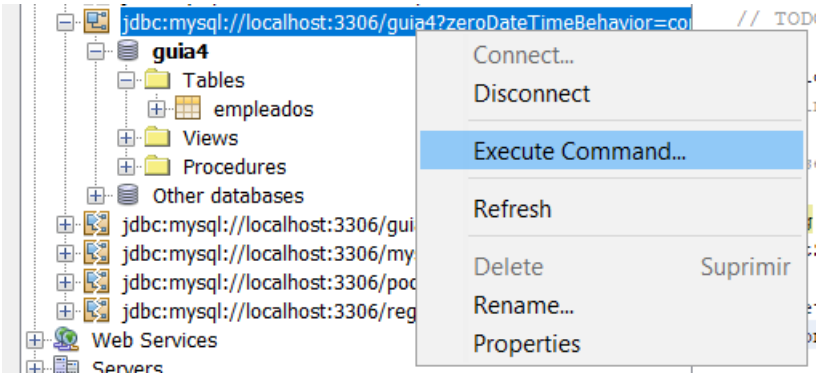


Se abre una nueva ventana donde se escribe los nombres de las columnas de la tabla, el tipo de dato, si tiene llave primaria, para agregar mas columnas hacer clic en **Addcolumn**, si quiere eliminar una columna hacer clic en **Remove.** Al terminar hacer clic en el botón **OK.**

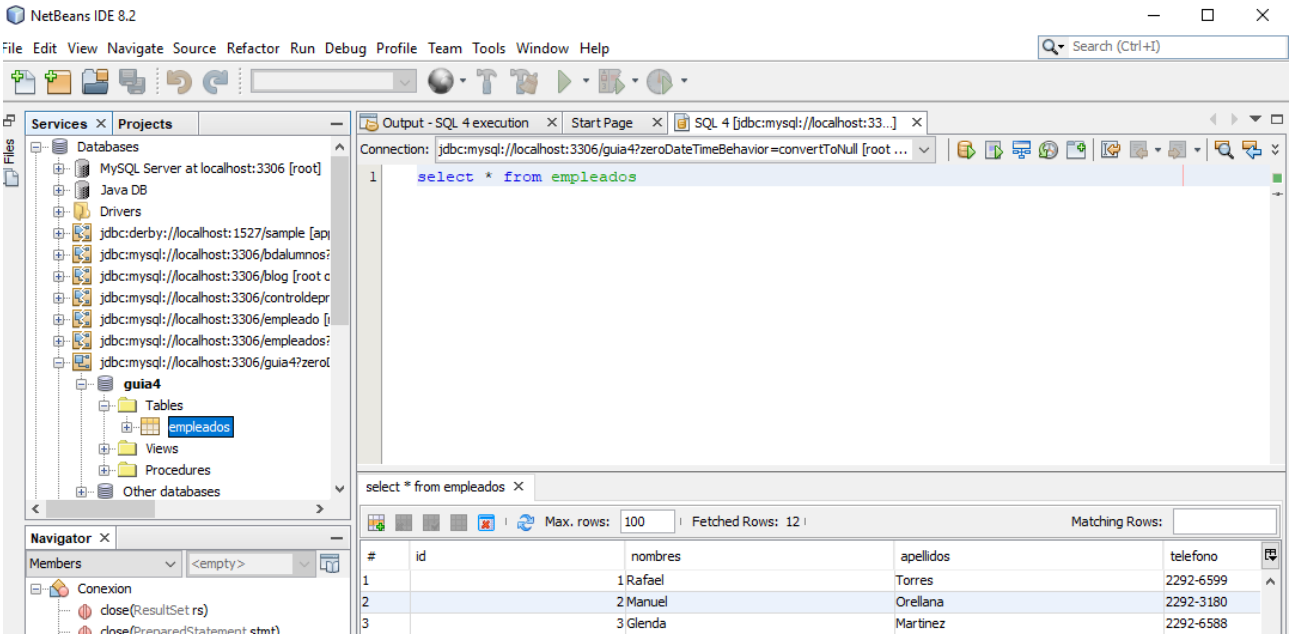


Para ejecutar un comando ya sea de inserción, eliminación, actualización y selección de información, seleccionar la opción **ExecuteCommand…**

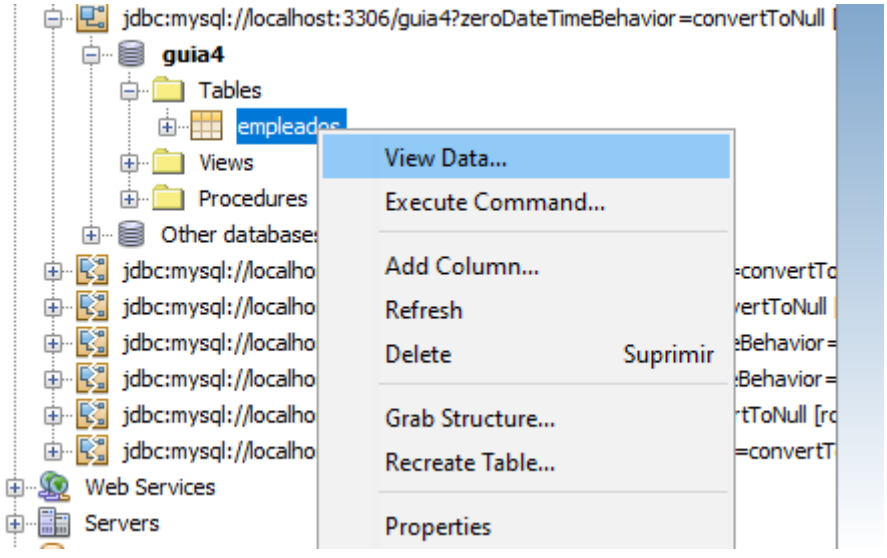
Se abrirá en Netbeans el área de edición de comandos o instrucciones SQL, donde digitamos los comandos. En la parte superior de la ventana se encuentran las transacciones y en la parte inferior muestra el resultado de la transacción. Y finalmente podemos manipular la información de una tabla



Para ejecutar la transacción y ver los resultados, se hace clic en el icono **Run SQL**.



También podemos ver los datos de una tabla seleccionando la opción **View Data,** la cual la seleccionamos haciendo clic derecho sobre el nombre de la tabla la cual queremos ver los datos.

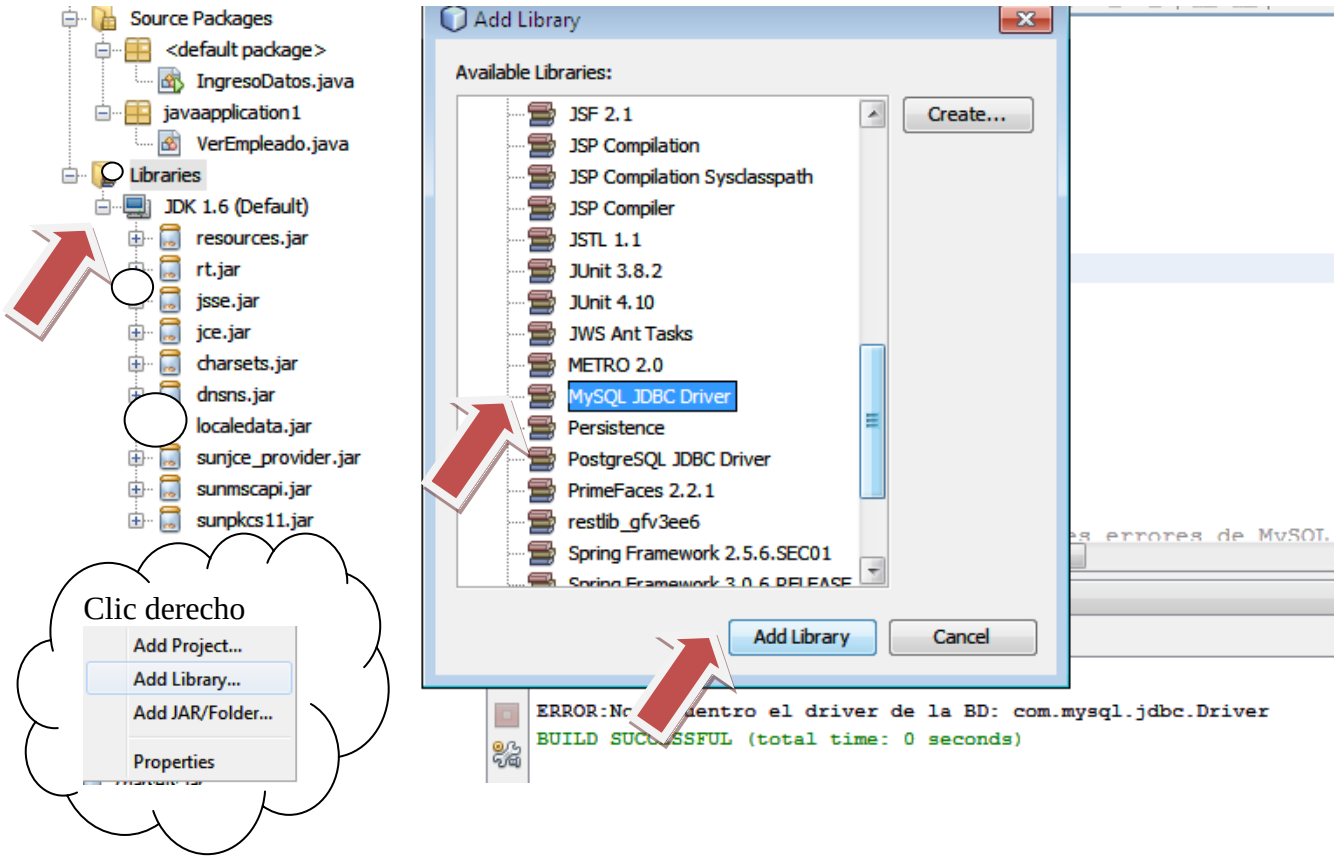


Acceder a los datos de una BD desde una aplicación Java

Agregue la librería de MySQL a las rutas antes mencionadas

C:\Archivos de programa\Java\jre1.6.0_05\lib\ext
Ó
C:\Program Files\Java\jdk1.6.0_25\jre\lib\ext

También puede agregar la librería directamente al proyecto



1. En el proyecto Guia4 debe crear una clase y colocar el nombre **VerEmpleado**, dentro del paquete "**sv.edu.udb.empleado**".
2. Digitar el siguiente código:

```
import java.sql.*;

public class VerEmpleado {

public VerEmpleado()
   {
// Se utiliza un try por los posibles errores de MySQL
try
      {
//obtenemos el driver de para mysql
Class.forName("com.mysql.jdbc.Driver");

// Se obtiene una conexión con la base de datos.
Connection conexion = DriverManager.getConnection (
"jdbc:mysql://localhost/Guia4","root", "rafael");
//          IMPORTANTE: EL CAMPO PASSWORD POR DEFECTO DEBE IR EN BLANCO,
//                DEBE ASIGNAR EL PASSWORD CORRECTO

// Permite ejecutar sentencias SQL sin parámetros
```

```
Statement s = conexion.createStatement();

// Contiene la tabla resultado de la pregunta SQL que se haya realizado
ResultSet rs = s.executeQuery ("select * from Empleados");

//Se recorre el ResultSet, mostrando por pantalla los resultados.

while (rs.next())
          {
/Podemos mostrar los datos de otra forma ver mas abajo e la guia.
System.out.println ("ID: "+rs.getInt(1)+
"\nNombre: "+rs.getString(2)+
"\nApellidos: "+rs.getString(3)+
"\nTelefono: "+rs.getString(4));
System.out.println("*********************************");
}


// Se cierra la conexión con la base de datos. NUNCA OLVIDE CERRARLA
conexion.close();
      }
catch (ClassNotFoundException e1) {
//Error si no puedo leer el driver de MySQL
                        System.out.println("ERROR:No encuentro el driver de la BD:
"+e1.getMessage());
      }
catch (SQLException e2) {
//Error SQL: login/passwdó sentencia sqlerronea
                        System.out.println("ERROR:Fallo en SQL: "+e2.getMessage());
      }
   }
/**
    * Método principal, instancia una clase PruebaMySQL
*
    * @paramargs the command line arguments
    */
public static void main(String[] args)
{
newVerEmpleado();
   }
   }
```

3. Ejecutar la aplicación y obtendremos los siguientes resultados:

4. Forma alternativa de mostrar los datos.

Para obtener los distintos atributos de la base se utilizarán, normalmente, los métodos getString(atributo), cuyo parámetro es una cadena con el nombre del atributo que queremos recuperar.

```
//Se recorre el ResultSet, mostrando por pantalla los resultados.
while (rs.next())
        {
System.out.println ("ID: "+rs.getInt("Codigo")+
"\nNombre: "+rs.getString("Nombres")+
"\nApellidos: "+rs.getString("Apellidos")+
"\nTelefono: "+rs.getString("Telefono"));
System.out.println("**********");
        }
```

Ejercicio 6: Ingresar datos desde una aplicación a una base de datos

1. En el proyecto Guia4_POO1 debe crear una clase y colocar el nombre IngresoDatos, dentro del paquete "**sv.edu.udb.empleado**".
2. Digitar el siguiente código.

```
import java.sql.*;
import javax.swing.JOptionPane;
import sv.edu.udb.util.* ;

public class IngresoDatos {
private int id;
private  String ids;
private  String nombre;
private  String apellido;
private  String telefono;

private  Connection conexion;
```

```
private  ResultSet rs;
private  Statement s;

public IngresoDatos()
   {
// Se utiliza un try por los posibles errores de MySQL
try
      {
//obtenemos el driver de para mysql
Class.forName("com.mysql.jdbc.Driver");

// Se obtiene una conexión con la base de datos.
conexion = DriverManager.getConnection (
"jdbc:mysql://localhost/Guia4","root", "rafael");
//          IMPORTANTE: EL CAMPO PASSWORD DEBE IR EN BLANCO POR DEFECTO,
//                DEBE ASIGNAR EL PASSWORD CORRECTO

// Permite ejecutar sentencias SQL sin parámetros
          s = conexion.createStatement();
//Metodo para ingresar valores
          ingreso();
s.executeUpdate("Insert into Empleados
values("+id+",\""+nombre+"\",\""+apellido+"\",\""+telefono+"\")");

JOptionPane.showMessageDialog(null,"Persona Ingresada Correctamente");
      }
catch (ClassNotFoundException e1) {
//Error si no puedo leer el driver de MySQL
          System.out.println("ERROR:No encuentro el driver de la BD: "+e1.getMessage());
System.exit(0);
      }
catch (SQLException e2) {
//Error SQL: login/passwdó sentencia sql erronea
          System.out.println("ERROR:Fallo en SQL: "+e2.getMessage());
System.exit(0);
      }
   }

public void ingreso()
   {
      ids=JOptionPane.showInputDialog("Ingrese el ID");
      id=Integer.parseInt(ids);
nombre=JOptionPane.showInputDialog("Ingrese el Nombre");
      apellido=JOptionPane.showInputDialog("Ingrese el Apellido");
telefono=JOptionPane.showInputDialog("Ingrese su Telefono");
         do {
               if ( MatchTelephone.compareTelephone(telefono) == TRUE)
               {
                  break;
               }else{
                  JOptionPane.showMessageDialog(null, "Numero de Telefono invalido");
                   telefono=JOptionPane.showInputDialog("Ingrese su Telefono");
               }
```

```
            } while (true);

}

public void mostrardatos()  throws SQLException{
rs = s.executeQuery ("select * from Empleados");
while (rs.next())
          {
JOptionPane.showMessageDialog(null,"ID: "+rs.getString("Codigo")+
"\nNombre: "+rs.getString("Nombres")+
"\nApellidos: "+rs.getString("Apellidos")+
"\nTelefono: "+rs.getString("Telefono")
               );
          }
   }

public void cierreconexion() throws SQLException{
// Se cierra la conexión con la base de datos.
               if (conexion != null){
                    conexion.close();
                 }
   }

public static void main(String[] args)  throws SQLException
{
IngresoDatosing=newIngresoDatos();
ing.mostrardatos();
ing.cierreconexion();
   }
   }
```

3. Crear la clase llamada **MatchTelephone** dentro del paquete "**sv.edu.udb.util**" y agregar el siguiente código, esta clase utilizara expresiones regulares para validar el formato de un número de teléfono.

```
package sv.edu.udb.util;

import static java.lang.Boolean.FALSE;
import static java.lang.Boolean.TRUE;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author Rafael Torres
 */
public class MatchTelephone {


//    public static void main(String[] args) {
```

```
//        compareTelephone("7123-4444");
//    }

    public static boolean compareTelephone (String telefono) {

        String expresion="(2|7)\\d{3}-\\d{4}";

        Pattern pat = Pattern.compile(expresion);
        Matcher mat = pat.matcher(telefono);
        if (mat.matches()) {
            System.out.println("SI");
            return TRUE;
        }

        return FALSE;
    } //Cierre del main

}
```

4. Para este ejemplo se realizara una análisis que nos permita comprender que no es lo mismo manejar datos “Vacios o Nulos”, en primer lugar crearemos la clase **Conexión** para que este en un paquete diferente “**sv.edu.udb.util**” y solo la invoquemos para futuros desarrollos.

```
package sv.edu.udb.util;

/**
 *
 * @author Rafael Torres
 */
import java.sql.*;

public class Conexion {
    private Connection conexion =null;
    private Statement s =null;
    private ResultSet rs=null;
    private String ingresoempleados="";

    //Contructor
    public Conexion() throws SQLException{
        try
        {
            //obtenemos el driver de para mysql
            Class.forName("com.mysql.jdbc.Driver");
            // Se obtiene una conexión con la base de datos.
            conexion = DriverManager.getConnection (
                "jdbc:mysql://localhost/guia4","root", "rafael");
            // Permite ejecutar sentencias SQL sin parámetros
            s = conexion.createStatement();

            System.out.println("Conexion Exitosa");

        }
        catch (ClassNotFoundException e1) {
```

```
            //Error si no puedo leer el driver de MySQL
                System.out.println("ERROR:No encuentro el driver de la BD:
"+e1.getMessage());
        }
    }

    //Metodo que permite obtener los valores del resulset
    public ResultSet getRs() {
        return rs;
    }


    //Metodo que permite fijar la tabla resultado de la pregunta
    //SQL realizada
    public void setRs(String sql)  {
        try {

            this.rs = s.executeQuery(sql);

        } catch (SQLException e2) {
            //Error SQL: login/passwd ó sentencia sql errónea
                System.out.println("ERROR:Fallo en SQL: "+e2.getMessage());
        }
    }

    //Metodo que recibe un sql como parametro que sea un update,insert.delete
    public void setQuery(String sql) throws SQLException {
        this.s.executeUpdate(sql);
    }

    //Metodo que cierra la conexion
    public void cerrarConexion() throws SQLException{
        conexion.close();
    }

}
```

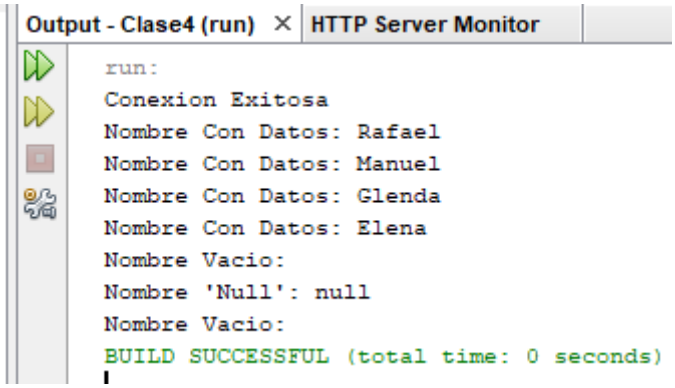
5. Crear la clase llamada "**InsertNulos**" dentro del paquete "**sv.edu.udb.nulos**" y agregar el siguiente código.

```
package sv.edu.udb.nulos;
import java.sql.ResultSet;
import java.sql.SQLException;
import sv.edu.udb.util.Conexion;

/**
 *
 * @author Rafael Torres
 */
public class InsertNulos {
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws SQLException {
        // TODO code application logic here

        Conexion con = new Conexion();
//        String sql = "insert into empleados values(7,'','Torres',null)";
//
       //con.setQuery(sql);

        String sql = "select nombres from empleados ";
        ResultSet rs ;

        con.setRs(sql);
        rs= con.getRs();

        String nombre;

        while (rs.next()){
            nombre=rs.getString(1);

            if (nombre == null){
                System.out.println("Nombre 'Null': " + nombre);
            }else if(nombre.equals("")){
                System.out.println("Nombre Vacio: " + nombre);
            }else{
                System.out.println("Nombre Con Datos: " + nombre);
            }


        }
        con.cerrarConexion();
    }

}
```

6. Ejecutar un par te sentencias update para la tabla **empleados.**
    - insert into empleados values(5,'','Torres',null);
    - insert into empleados values(6,null,Rodriguez,null);

7. Ejecutar la clase **InsertNulos** para ver un resultado como el de la siguiente imagen



IV. EJERCICIOS COMPLEMENTARIOS

- **Crear las siguientes tablas en MySql:**

Tabla alumno

Nombre Columna	Tipo de dato
Cod_alumno	int primary key
Nombre	varchar(80)
Apellido	varchar(80)
Edad	int
Direccion	varchar(100)

Tabla materia

Nombre Columna	Tipo de dato
Cod_materia	int primary key
Nombre	varchar(25)
Descripción	varchar(100)

Tabla alumno_materia

Nombre Columna	Tipo de dato
Cod_alumno	int (llave foránea)
Cod_materia	int (llave foránea)

Crear el CRUD a las 3 tablas anteriores:

- Alumno
- Materia
- Alumno_materia

Reportes

1. Crear una aplicación donde muestre la información de cada una de las tablas anteriores, para ello crear tres métodos diferentes que serán invocados según la necesidad.
2. Mostrar las materias que está cursando un alumno específico.
   Tip: **Usará la cláusula where en la sentencia SQL**

Nota: Para cada ejercicio solicitado utilizar JoptionPane y validar los datos. Debe usar una clase pre-fabricada para validar.