UNIDAD 9. Acceso a Base de Datos desde Java

9.1 Open Database Connectivity (ODBC)

Open Database Connectivity (ODBC) es un estándar desarrollado por Microsoft, el cual permite conectarse a un DBMS (Database Management Systems). Este API (Application Programming Interface) es independiente del lenguaje de programación y del manejador de base de datos, ambos deben de ser compatibles con el ODBC. El ODBC es el intermediario entre las aplicaciones y la base de datos, permite usar el mismo código para trabajar con cualquier base de datos con la restricción de usar el estándar ANSI SQL en los comandos.

9.2 API JDBC (Java DataBase Connectivity)

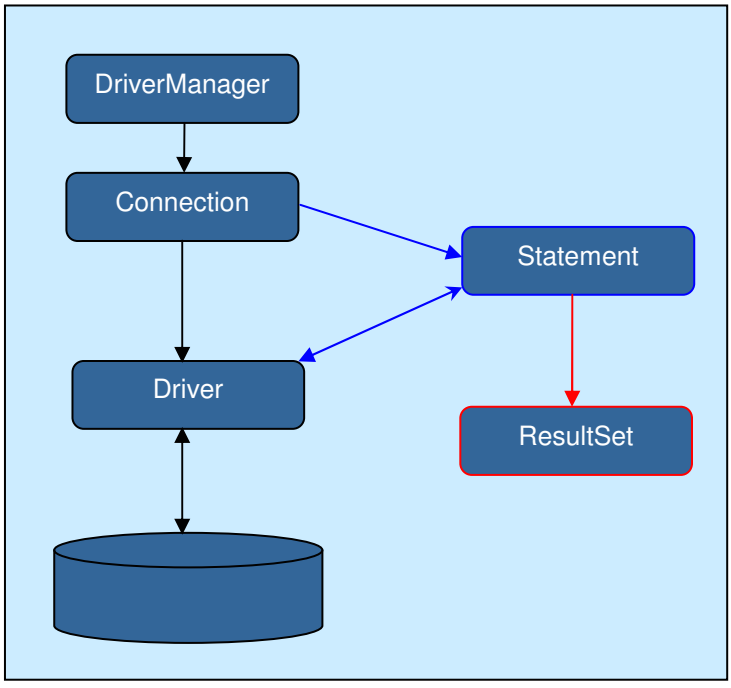
El API JDBC es el estándar de conexión entre el lenguaje de programación Java y un amplio rango de bases de datos. Este API hace posible la conexión con un DBMS (DataBase Management System), permite enviar enunciados SQL y procesar los resultados.

El uso del API JDBC hace posible llevar a cabo lo siguiente:

- Establecer una conexión con una base de datos o acceder a cualquier fuente de datos tabular
- Enviar enunciados SQL.
- Procesar los resultados

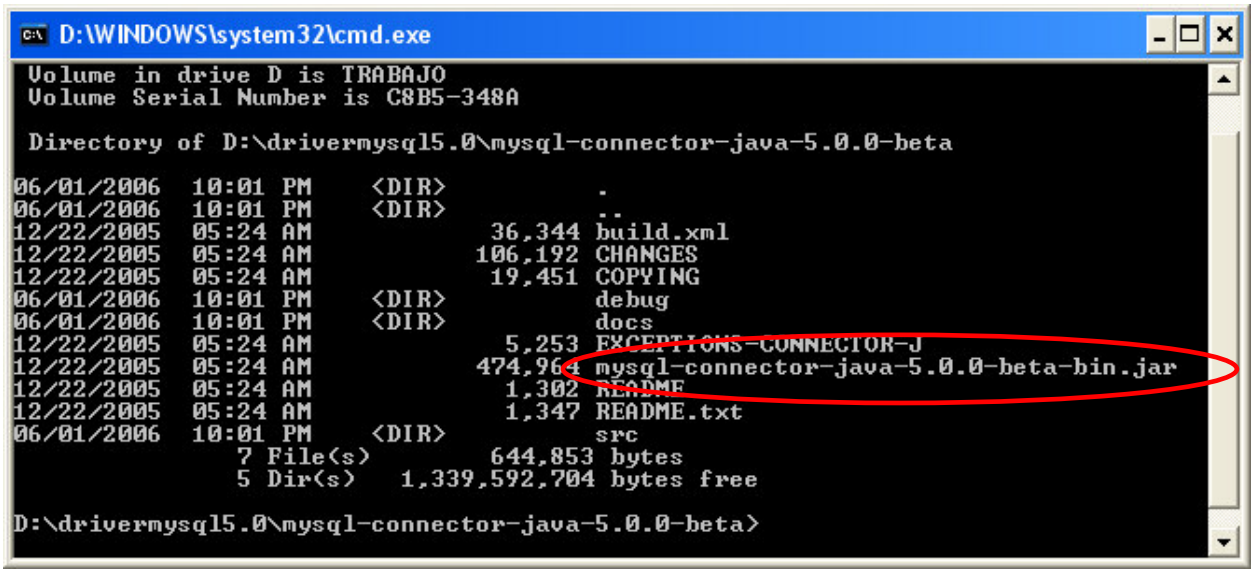
En la siguiente figura se ven las operaciones que pueden realizarse en una base de datos. Cada uno de los rectángulos representa una clase o interfaz de JDBC que tiene un rol en el acceso a una base de datos relacional. Todo el trabajo con el JDBC inicia con la clase DriveManager, que es la que establece las conexiones con las fuentes de datos, mediante el driver JDBC.

El objeto DriverManager crea una conexión con la base de datos mediante el driver. Los comandos SQL (Statements) se envían a la base de datos a través de la conexión establecida. Cuando el comando genera resultados estos se almacenan en un objeto ResultSet.

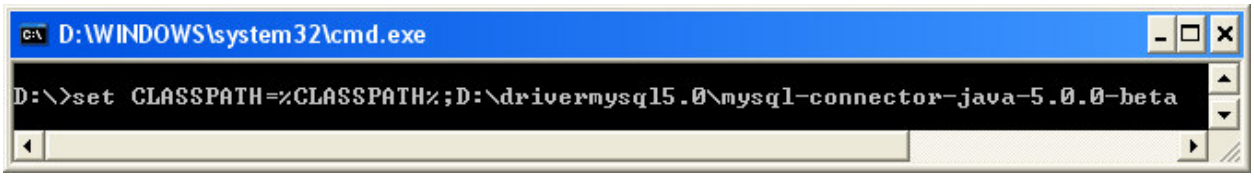


9.3 Instalación del driver Jconnector

Antes de establecer cualquier conexión se debe instalar el driver Java-Mysql. El Driver para Mysql se llama connector/J, y está disponible en ***http://dev.mysql.com/downloads/connector/j/5.0.html.*** El archivo zip debe desempaquetarse en un directorio, por ejemplo en c:\mysqldriver. Al desempaquetar el driver se crea el directorio ***mysql-connector-java-5.0.0-beta*** en el que se encontrarán los siguientes archivos:



ahora para que Java localice la clase que implementa el driver esto es la clase com.mysql.jdbc.Driver, debe agregar a la variable CLASSPATH la ruta donde está el archivo ***mysql-connector-java-5.0.0-beta-bin.jar*** . Lo anterior se hace con la línea:



En ocasiones el procedimiento anterior no es suficiente para que java ubique el driver, en este caso otra solución sería copiar el archivo ***mysql-connector-java-5.0.0-beta-bin.jar*** al directorio donde se haya instalado el JRE: ***D:\Program Files\Java\jre1.5.0_06\lib\ext***

9.4 Pasos para establecer la conexión java-dbms

El establecimiento de la conexión entre la aplicación java y la base de datos implica los siguientes pasos:

Paso1: Registro del driver, en este caso el driver para mysql. Dado que se va a trabajar con mysql se requiere registrar el driver para mysql, esto se hace con el método Class.forName.

```
String driver = "com.mysql.jdbc.Driver";
Class.forName( driver ).newInstance();
```

Paso 2: Construcción de la URL Una vez que el driver ha sido cargado entonces se puede hacer la conexión con el método DriverManager.getConnection. Este método requiere los parámetros url, login y password. El url (Uniform Resource Locator), especifica la forma cómo se va a conectar a conectar la base de datos, el servidor en el que se encuentra y el nombre de la base de datos. Para este curso la conexión se va a realizar usando el puente jdbc-mysql, la base de datos estará ubicada en la misma máquina que la aplicación, por lo tanto es localhost y finalmente el nombre de la base de datos es la que Mysql provee por defaul: test.

```
driver=new String("com.mysql.jdbc.Driver");
url=new String("jdbc:mysql://localhost/test");
login=new String("root");
password=new String(" ");
conexion=DriverManager.getConnection(url,login,password);
```

Las operaciones de registro del driver y establecimiento de la conexión generan las excepciones java.lang.ClassNotFoundException, java.lang.InstantiationException y java.sql.SQLException por lo que es necesario atraparlas por medio del bloque try-catch. El siguiente segmento de código sirve para probar una conexión a una base de datos.

```
import java.sql.*;

class BasedeDatos{
   String driver,url,login,password;
   Connection conexion;

   BasedeDatos(){
       driver="com.mysql.jdbc.Driver";
       url=new String("jdbc:mysql://localhost/test");
       login=new String("root");
       password=new String(" ");
       try{ Class.forName(driver).newInstance();
            conexion=DriverManager.getConnection(url,login,password);
            System.out.println("Base de Datos ok...");
       }
       catch (Exception exc){
          System.out.println("Error al tratar de abrir la base de Datos");
      }
    }
}

class PruebaBasedeDatos{
            public static void main(String args[]){
                        BasedeDatos bd =new BasedeDatos();
            }
}
```

9.5 Envío de comandos SQL al DBMS

Statement es la clase usada para ejecutar un enunciado SQL y regresar los resultados que el enunciado produce. Por default, solamente un objeto **ResultSet** por enunciado puede estar abierto, sin embargo, si la lectura de un objeto **ResultSet** es alternada con la lectura de otro objeto, cada uno debe ser generado por diferentes objetos **Statement.**

9.5.1 Métodos de la clase Statement

Método	Descripción
boolean execute(String sql)	Ejecuta el enunciado SQL especificado.
ResultSet executeQuery(String sql)	Ejecuta el enunciado SQL indicado, regresa un solo objeto ResultSet.
int ExecuteUpdate(String sql)	Ejecuta el enunciado SQL especificado, el cual puede ser INSERT, UPDATE, DELETE un enunciado SQL que no regresa resultados.

9.6 Creación de Base de Datos

Una vez establecida una conexión con el DBMS es posible crear una base de datos nueva con el enunciado CREATE DATABASE. La opción IF NOT EXIST evita que se produzca un error si la base de datos ya existe.

```
import java.sql.*;
class MysqlCreaBase{
   Connection conexion;
    void conectar(){
   try {    String driver = "com.mysql.jdbc.Driver";
            System.out.println( "\n=> loading driver:" );
            Class.forName(driver).newInstance();
            System.out.println( "OK" );
            String url = "jdbc:mysql://localhost/test";
            System.out.println( "\n=> connecting:" );
            conexion=DriverManager.getConnection( url, "root", "ginger" );
            System.out.println( "OK" );
    }
   catch( Exception x ) {
      System.err.println( x );
   }
  }

void crearBase(){
            String crear="CREATE DATABASE IF NOT EXISTS alumnos; ";
            try{  Statement stmt=conexion.createStatement();
                  stmt.execute(crear);
            }
            catch (java.sql.SQLException exc){
                        System.out.println("Error ");
            }
}

 class PruebaMysqlCreaBase{
   public static void main(String args[]) {
            MysqlCreaBase mysql=new MysqlCreaBase();
            mysql.conectar();
            mysql.crearBase();
  }
 }
```

9.7 Creación de Tablas

Una vez que ya existe una conexión con una base de datos, es posible crear una tabla usando el enunciado SQL CREATE. El enunciado se envía usando el método executeUpdate. Una operación de este tipo genera una excepción java.sq.SQLException, por ejemplo si la tabla ya existe. En el ejemplo el uso de IF EXISTS evita que se produzca un error si la tabla ya existe.

```
import java.sql.*;

class MysqlCrearTablas{

   Connection conexion;

  void conectar(){

   try {
      String driver = "com.mysql.jdbc.Driver";
      System.out.println( "\n=> loading driver:" );
      Class.forName(driver).newInstance();
      System.out.println( "OK" );

      String url = "jdbc:mysql://localhost/test";
      System.out.println( "\n=> connecting:" );
      conexion=DriverManager.getConnection( url, "root", "ginger" );
      System.out.println( "OK" );
    }

   catch( Exception x ) {
      System.err.println( x );
   }
  }

void crearTablas(){
    String crear="CREATE TABLE IF NOT EXISTS datos (nombre VARCHAR(40), edad INTEGER, sueldo
                                                                                FLOAT); ";
     try{Statement stmt=conexion.createStatement();
         stmt.execute(crear);
     }
     catch (java.sql.SQLException exc){
          System.out.println("Error ");
       }
    }
}

class PruebaMysqlCrearTablas{
     public static void main(String args[]) {

Mysql mysql=new Mysql();
            mysql.conectar();
            mysql.crearTablas();
  }
 }
```

9.8 Inserción de datos en una tabla.

El comando SQL para agregar datos a una tabla ya existente es INSERT. Se envía a MySql de la misma forma que el comando anterior, primero se conforma el enunciado en una cadena y después se envía a través de un objeto Statement. Es importante hacer notar que de acuerdo a la sintaxis SQL los valores String que se vayan a insertar deben estar encerrados entre apóstrofes.

String enun="INSERT INTO datos VALUES (' " + nombre + "'," + dias +"," +sueldo +");";

En el enunciado los nombre, dias y sueldo serán sustituidos por sus valores para insertarse en la tabla.

9.8.1 Ejemplo de Inserción de datos en una tabla

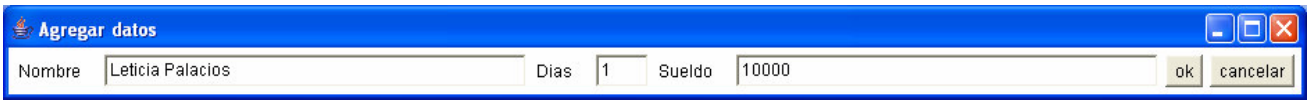
```
import java.awt.*;
import java.sql.*;
import java.awt.event.*;

class Persona{
   String nombre;
   double sueldo;
   int    dias;
}
class MysqlAgregar{
            Connection conexion;
            void conectar(){
               try {      String driver = "com.mysql.jdbc.Driver";
                          System.out.println( "\n=> loading driver:" );
                          Class.forName(driver).newInstance();
                          String url = "jdbc:mysql://localhost/test";
                          System.out.println( "\n=> connecting:" );
                          conexion=DriverManager.getConnection( url, "root", "ginger" );
                          System.out.println( "OK" );
                        }
               catch( Exception x ) {
                          System.err.println( x );
               }
              }

            void agregarDatos(){      Persona datos=new Persona();
                                      Formas forma=new Formas();
                                      forma.formaAgregar();
            }

            void grabarDatos(Persona datos){
            String comando="INSERT INTO datos VALUES (' " + datos.nombre + "'," + datos.dias +" ,"
                                                                              +datos.sueldo +");";
                                    try{      Statement stmt=conexion.createStatement();
                                              stmt.executeUpdate(comando);
                                              System.out.println("Registro agregado!");
                                    }
                                    catch( java.sql.SQLException er){
                                               System.out.println("No se pudo realizar la operacion");
                                    }
            }
```

continua Ejemplo 9.8.1 Inserción de datos en una tabla …

```
class Formas{
  TextField tNombre,tDias,tSueldo;
  Button ok, cancelar;
  protected Frame v;
  EventosBoton monitorBoton;
  ManejoDeEventosFrame monitorFrame;

  void formaAgregar(){
                        v=new Frame("Agregar datos");
                        v.setLayout(new FlowLayout() );
                        v.add(new Label("Nombre") );
                        v.add(tNombre=new TextField(40) );
                        v.add(new Label("Dias") );
                        v.add(tDias=new TextField("0",2) );
                        v.add(new Label("Sueldo") );
                        v.add(tSueldo=new TextField("0",40) );
                        v.add(ok=new Button("ok"));
                        v.add(cancelar=new Button("cancelar") );
                        monitorBoton=new EventosBoton();
                        ok.addActionListener(monitorBoton);
                        cancelar.addActionListener(monitorBoton);
                        v.addWindowListener(monitorFrame =new ManejoDeEventosFrame() );
                        v.pack();
                        v.setVisible(true);
  }

  void recuperaDatos(){
            Persona datos=new Persona();
            datos.nombre=tNombre.getText();
            try{
                        datos.dias=new Integer(tDias.getText() ).intValue();
            }
            catch( Exception exc){
                        System.out.println("Error al introducir los datos");
                        tDias.setText("0");datos.dias=0;
            }

            try{
                        datos.sueldo=new Double(tSueldo.getText() ).doubleValue();
            }
            catch( Exception exc){
                        System.out.println("Error al introducir los datos");
                          tSueldo.setText("0");datos.sueldo=0;
            }
            if (datos.nombre.length()>0) grabarDatos(datos);
            else
              System.out.println("Nombre vacio");
  }

  void limpiaTexto(){
            tNombre.setText(" ");
            tDias.setText("0");
            tSueldo.setText("0");
  }
```

continua Ejemplo 9.8.1 Inserción de datos en una tabla …

```
  class EventosBoton implements ActionListener{
      public void actionPerformed(ActionEvent evento){
                  Object fuente=evento.getSource();
           if(fuente==ok) recuperaDatos();
                                 if(fuente==cancelar) limpiaTexto();
       }
    }


  class ManejoDeEventosFrame extends WindowAdapter{
           public void windowClosing(WindowEvent evento){
          if (evento.getSource()==v){
                                          v.setVisible(false);
              v.dispose();
          }
    }
  }
}
}

class PruebaMysqlAgregar{
  public static void main(String args[]) {
           MysqlAgregar mysql=new MysqlAgregar();
           mysql.conectar();
           mysql.agregarDatos();
  }
 }
```



Pantalla generada por el ejemplo 9.8.1

9.9 Consulta general

Los enunciados SQL que regresan resultados se envían al DBMS a través del comando executeQuery. El comando SELECT * FROM datos; regresa todos los registros que están en la tabla y se depositan en un objeto de la clase ResultSet. Para visualizar todos los registros se implementa un ciclo que recorre el conjunto de resultado, extrae los valores de cada campo y avanza al siguiente registro.

9.9.1 Clase ResultSet

La clase ResulSet es un tabla de datos representando el resultados de una búsqueda en una base de datos .

9.9.2 Métodos de la clase ResultSet

Método	Descripción
void afterLast()	Mueve el cursor después del último renglón del objeto ResultSet.
boolean next()	Avanza al siguiente registro dentro de un objeto ResultSet. Regresa falso si la operación no es posible.
void beforeFirst()	Mueve el cursor antes del primer renglón del objeto ResultSet.
boolean first	Retrocede un renglón dentro de un objeto ResultSet. Regresa falso si la operación no es posible.
String getString(int n)	Regresa la cadena que se encuentra en el campo n del registro actual.
int getInt(int n)	Regresa el valor entero que está en el campo n del registro actual.
float getFloat(int n)	Regresa el valor flotante que está en el campo n del registro actual.

9.9.3 Implementación de consulta general en una lista desplazable

```
import java.awt.*;
import java.sql.*;
import java.awt.event.*;

class Persona{
   String nombre;
   double sueldo;
   int    dias;
}
class MysqlAgregar{
    Connection conexion;
     void conectar(){   try {  String driver = "com.mysql.jdbc.Driver";
                               System.out.println( "\n=> loading driver:" );
                               Class.forName(driver).newInstance();
                               String url = "jdbc:mysql://localhost/test";
                               System.out.println( "\n=> connecting:" );
                               conexion=DriverManager.getConnection( url, "root", "ginger" );
                                System.out.println( "OK" );
                          }
                         catch( Exception x ) {   System.err.println( x );
                         }
                        }

void consultaGeneral(){
                      Formas forma=new Formas();
                      forma.formaMostrar();
}
```

continua 9.9.3 Implementación de consulta general en una lista desplazable

```
class Formas{
  Frame v;
  ManejoDeEventosFrame monitorFrame;

          void formaMostrar(){ String n; double s;  int d;
                              v=new Frame();
                              v.setLayout(new FlowLayout() );
                              v.setTitle("Mostrando Datos");
                              v.addWindowListener(monitorFrame= new ManejoDeEventosFrame() );
                              java.awt.List todos=new java.awt.List(5);
                              try{Statement stmt=conexion.createStatement();
                                        ResultSet result= stmt.executeQuery("SELECT * from datos;");
                                         while(result.next() ){
                                         n=result.getString(1);
                                         d=result.getInt(2);
                                         s=result.getFloat(2);
                                         todos.add(n+" "+ String.valueOf(d)+" "+ String.valueOf(s) );
                                        }
                              }
                              catch(java.sql.SQLException er){
                                         System.out.println("No se pudo realizar la operacion");
                              }
                              v.add(todos);
                              v.pack();
                              v.setVisible(true);
                    }

            class ManejoDeEventosFrame extends WindowAdapter{
                    public void windowClosing(WindowEvent evento){
                                        if (evento.getSource()==v){ v.setVisible(false);
                                                  v.dispose();
                                        }
                    }
             }
          }
}
class PruebaMysqlMostrar{
  public static void main(String args[]) {
          MysqlAgregar mysql=new MysqlAgregar();
          mysql.conectar();
          mysql.consultaGeneral() ;
  }
 }
```

En el ejemplo anterior, los registros obtenidos de la base de datos se muestran en una lista desplazable.

9.9.4 Implementación de consulta general mostrando un solo registro por ventana.

```
import java.awt.*;
import java.sql.*;
import java.awt.event.*;
class Persona{    String nombre;
                  double sueldo;
                   int    dias;
}

class MysqlMostrar2{   Connection conexion;
  void conectar(){   try {     String driver = "com.mysql.jdbc.Driver";
                               Class.forName(driver).newInstance();
                               String url = "jdbc:mysql://localhost/test";
                               System.out.println( "\n=> connecting:" );
                               conexion=DriverManager.getConnection( url, "root", "ginger" );
                               System.out.println( "OK" );
                             }
                    catch( Exception x ) {  System.err.println( x );
                    }
 }

 void consultaGeneral(){          Formas forma=new Formas();
                                  forma.formaMostrar();
 }

class Formas{
  TextField tNombre,tDias,tSueldo;
  Button ok, cancelar,siguiente,anterior;
  Frame v;
  EventosBoton monitorBoton;
  ManejoDeEventosFrame monitorFrame;
  ResultSet result;

void formaMostrar(){   String n=""; double s=0;     int d=0;
                    v=new Frame("Mostrando Registros");
                    v.setLayout(new FlowLayout() );
                    v.add(new Label("Nombre") );     v.add(tNombre=new TextField(40) );
                    v.add(new Label("Dias") );       v.add(tDias=new TextField("0",2) );
                    v.add(new Label("Sueldo") );     v.add(tSueldo=new TextField("0",40) );
                    v.add(anterior=new Button("anterior"));    v.add(siguiente=new Button("siguiente") );
                    monitorBoton=new EventosBoton();
                    v.addWindowListener(monitorFrame =new ManejoDeEventosFrame() );

                   try{      Statement stmt=conexion.createStatement();
                             result= stmt.executeQuery("SELECT * from datos;");
                   }

                  catch (java.sql.SQLException er){
                             System.out.println("No se pudo realizar la operacion");
                  }
                  siguiente.addActionListener(monitorBoton);
                  anterior.addActionListener(monitorBoton);
                  v.pack();
                  v.setVisible(true);
   }
```

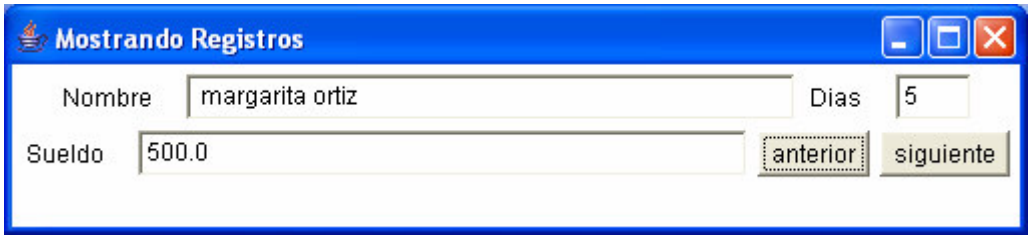
continua 9.9.5 Implementación de consulta general mostrando un solo registro por ventana.

```
void registroAnterior(){
            String n="";
            double s=0;
            int d=0;
            try{
             result.previous();
             n=result.getString(1);
             d=result.getInt(2);
             s=result.getFloat(3);
            }

             catch(java.sql.SQLException er){
                    System.out.println("No hay registro anterior");
        }

        tNombre.setText(n);
        tDias.setText(new Integer(d).toString()  );
        tSueldo.setText(new Double(s).toString() );
    }


    void registroSiguiente(){
            String n="";
            double s=0;
            int d=0;
            try{
                    result.next();
                    n=result.getString(1);
                    d=result.getInt(2);
                    s=result.getFloat(3);
           }

           catch(java.sql.SQLException er){
                System.out.println("No hay registro siguiente");
           }

        tNombre.setText(n);
        tDias.setText(new Integer(d).toString()  );
        tSueldo.setText(new Double(s).toString() );
     }

     class EventosBoton implements ActionListener{
                                public void actionPerformed(ActionEvent evento){
                                                Object fuente=evento.getSource();
                                                if  (fuente==anterior)
                                                        registroAnterior();
                                                if  (fuente==siguiente)
                                                        registroSiguiente();

                                }
       }
```

continua 9.9.5 Implementación de consulta general mostrando un solo registro por ventana.

```
class ManejoDeEventosFrame extends WindowAdapter{
                        public void windowClosing(WindowEvent evento){
                                if (evento.getSource()==v){
                                        v.setVisible(false);
                                        v.dispose();
                                }
                        }
                }
        }
}

class PruebaMysqlMostrar2{
  public static void main(String args[]) {
        MysqlMostrar2 mysql=new MysqlMostrar2();
        mysql.conectar();
        mysql.consultaGeneral() ;
  }
 }
```

En el ejemplo 9.9.5 cada registro se muestra en una ventana, se puede avanzar o retroceder en el conjunto de datos a través de los botones anterior y siguiente.



9.9.6 Consulta por campo específico

En los ejemplos anteriores la consulta tenía como resultado todos los registros contenidos en la tabla, pero si se desea ver solamente los registros que cumplan con cierta condición es necesario modificar el enunciado SQL para limitar los resultados. En el ejemplo, los resultados van a ser aquellos registros donde el contenido del campo nombre sea igual al campo texto buscar.

9.9.7 Ejemplo Consulta por campo específico

```
import java.awt.*;
import java.sql.*;
import java.awt.event.*;

class Persona{
   String nombre;
   double sueldo;
   int    dias;
}
class MysqlConsulta{    Connection conexion;
          void conectar(){
                    try {      String driver = "com.mysql.jdbc.Driver";
                               Class.forName(driver).newInstance();
                               String url = "jdbc:mysql://localhost/test";
                               System.out.println( "\n=> connecting:" );
                               conexion=DriverManager.getConnection( url, "root", "ginger" );
                               System.out.println( "OK" );
                              }
                    catch( Exception x ) {
                               System.err.println( x );
                    }
           }

          void consulta(){ Formas forma=new Formas();
                           forma.busqueda();
          }

class Formas{
  TextField tNombre,tDias,tSueldo,tBuscar;
  Button buscar,siguiente,anterior;
  Frame v, vCaptura;
  EventosBoton monitorBoton;
  ManejoDeEventosFrame monitorFrame;
  ResultSet result;

 void busqueda(){vCaptura=new Frame("Consulta por nombre");
                    vCaptura.setLayout(new FlowLayout() );
                    vCaptura.add(new Label("Nombre") );
                    vCaptura.add(tBuscar= new TextField (40) );
                    vCaptura.add(buscar=new Button("buscar"));
                    monitorBoton=new EventosBoton();
                    vCaptura.addWindowListener(monitorFrame =new ManejoDeEventosFrame() );
                    buscar.addActionListener(monitorBoton);
                    vCaptura.pack();
                    vCaptura.setVisible(true);
   }
```

continua 9.9.7 Implementación de consulta por campo específico

```
void formaConsulta(){
            String n="";
            double s=0;
            int d=0;
            v=new Frame("Mostrando Registros");
            v.setLayout(new FlowLayout() );
            v.add(new Label("Nombre") );
            v.add(tNombre=new TextField(40) );
            v.add(new Label("Dias") );
            v.add(tDias=new TextField("0",2) );
            v.add(new Label("Sueldo") );
            v.add(tSueldo=new TextField("0",40) );
            v.add(anterior=new Button("anterior"));
            v.add(siguiente=new Button("siguiente") );
            monitorBoton=new EventosBoton();
            v.addWindowListener(monitorFrame =new ManejoDeEventosFrame() );
            try{
            Statement stmt=conexion.createStatement();
            result= stmt.executeQuery("SELECT * FROM datos WHERE nombre=' "+tBuscar.getText() + "';");
            System.out.println("Buscando a " +tBuscar.getText() );
            if (result.next()) {            n=result.getString(1);
                                            d=result.getInt(2);
                                            s=result.getFloat(3);
                                            tNombre.setText(n);
                                            tDias.setText(new Integer(d).toString()  );
                                            tSueldo.setText(new Double(s).toString() );
                                            siguiente.addActionListener(monitorBoton);
                                            anterior.addActionListener(monitorBoton);
                                            v.pack();
                                            v.setVisible(true);
                        }
            }

       catch(java.sql.SQLException er){       System.out.println("No se pudo realizar la operacion");
       }
     }

void registroAnterior(){  String n="";   double s=0;    int d=0;
            try{result.previous();
                        n=result.getString(1);
                        d=result.getInt(2);
                        s=result.getFloat(3);
            }
            catch(java.sql.SQLException er){
                        System.out.println("No hay registro anterior");
            }
            tNombre.setText(n);
            tDias.setText(new Integer(d).toString()  );
            tSueldo.setText(new Double(s).toString() );
}
```

continua 9.9.7 Implementación de consulta por campo específico

```
void registroSiguiente(){ String n="";       double s=0;       int d=0;
          try{       result.next();
                     n=result.getString(1);
                     d=result.getInt(2);
                     s=result.getFloat(3);
          }
          catch(java.sql.SQLException er){ System.out.println("No hay registro siguiente");
          }
          tNombre.setText(n);
          tDias.setText(new Integer(d).toString()  );
          tSueldo.setText(new Double(s).toString() );
          }
          class EventosBoton implements ActionListener{
                    public void actionPerformed(ActionEvent evento){
                                        Object fuente=evento.getSource();
                                        if(fuente==anterior) registroAnterior();
                                        if(fuente==siguiente) registroSiguiente();
                                       if (fuente==buscar) formaConsulta();
                              }
          }
          class ManejoDeEventosFrame extends WindowAdapter{
                    public void windowClosing(WindowEvent evento){
                              if (evento.getSource()==vCaptura){
                                        if  (v!=null){    v.setVisible(false);
                                                  v.dispose();
                                        }
                                        vCaptura.setVisible(false);
                                        vCaptura.dispose();
                              }
                              if (evento.getSource()==v){ v.setVisible(false);
                                                  v.dispose();
                              }
                    }
               }
        }
}
class PruebaMysqlConsulta{
  public static void main(String args[]) {
          MysqlConsulta mysql=new MysqlConsulta();
          mysql.conectar();
          mysql.consulta() ;
    }
 }
```

Ventanas del ejemplo 9.9.7

9.10 Actualización de Registros.

La modificación de los datos ya almacenados en una tabla se hace con el comando SQL UPDATE. Por ser un comando que afecta a la base de datos, se envia a través de un objeto Statement con el método **executeUpdate.** El comando Update tiene la siguiente sintaxis:

UPDATE <tabla> SET <campo1=nuevoValor, campo2=nuevoValor, ...campoN=nuevoValor> WHERE <campo=criterio>;

Cabe hacer notar que el comando UPDATE actualizan solamante los campos que se incluyen en el enunciado, los campos que no se incluyan permanecen con el valor original.

9.10.1 Ejemplo de Actualización de Registros

```
import java.awt.*;
import java.sql.*;
import java.awt.event.*;

class Persona{
   String nombre;
   double sueldo;
   int    dias;
}
class MysqlCambios{
            Connection conexion;

void conectar(){
   try {
               String driver = "com.mysql.jdbc.Driver";
               System.out.println( "\n=> loading driver:" );
               Class.forName(driver).newInstance();
               String url = "jdbc:mysql://localhost/test";
               System.out.println( "\n=> connecting:" );
               conexion=DriverManager.getConnection( url, "root", "ginger" );
               System.out.println( "OK" );
              }
   catch( Exception x ) {
               System.err.println( x );
   }
 }

void cambios(){
        Formas forma=new Formas();
        forma.busqueda();
}

void grabarDatos(Persona datos){
  String comando="UPDATE datos SET sueldo="+datos.sueldo +",edad="+datos.dias+
                                        " WHERE nombre='"+ datos.nombre + "';";
    try{
            Statement stmt=conexion.createStatement();
            stmt.executeUpdate(comando);
   }
   catch( java.sql.SQLException er){
            System.out.println("No se pudo realizar la operacion");
   }
```

continua 9.10.1 Ejemplo de Actualización de Registros

```
class Formas{   TextField tNombre,tDias,tSueldo,tBuscar;
                Button buscar,modificar;
                Frame v, vCaptura;
                EventosBoton monitorBoton;
                ManejoDeEventosFrame monitorFrame;
                ResultSet result;

        void busqueda(){ vCaptura=new Frame("Modificaciones");
                         vCaptura.setLayout(new FlowLayout() );
                         vCaptura.add(new Label("Nombre") );
                         vCaptura.add(tBuscar= new TextField (40) );
                         vCaptura.add(buscar=new Button("buscar"));
                         monitorBoton=new EventosBoton();
                         vCaptura.addWindowListener(monitorFrame=new ManejoDeEventosFrame() );
                         buscar.addActionListener(monitorBoton);
                         vCaptura.pack();
                         Captura.setVisible(true);
        }

        void formaCambios(){
                String n=""; double s=0; int d=0;
                 v=new Frame("Modificaci\u00f3n de Registros");
                 v.setLayout(new FlowLayout() );
                 v.add(new Label("Nombre") );
                 v.add(tNombre=new TextField(40) );
                 v.add(new Label("Dias") );
                 v.add(tDias=new TextField("0",2) );
                 v.add(new Label("Sueldo") );
                 v.add(tSueldo=new TextField("0",40) );
                 modificar=new Button("Modificar");
                 v.add(modificar);
                 monitorBoton=new EventosBoton();
                 v.addWindowListener(monitorFrame =new ManejoDeEventosFrame() );
                 try{    Statement stmt=conexion.createStatement();
                         result=stmt.executeQuery("SELECT * FROM datos WHERE nombre=' " +
                                                                  tBuscar.getText() + "';");
                         if (result.next()) {        n=result.getString(1);
                                                     d=result.getInt(2);
                                                     s=result.getFloat(3);
                                                     tNombre.setText(n);
                                                     tNombre.setEditable(false);
                                                     tDias.setText(new Integer(d).toString()  );
                                                     tSueldo.setText(new Double(s).toString() );
                                                     modificar.addActionListener(monitorBoton);
                                                     v.pack();
                                                     v.setVisible(true);
                                 }
                         }

                         catch(java.sql.SQLException er){ System.out.println("No se pudo realizar");
                         }

                }
```

continua 9.10.1 Ejemplo de Actualización de Registros

```
void recuperaDatos(){
   Persona datos=new Persona();
   datos.nombre=tNombre.getText();
   try{       datos.dias=new Integer(tDias.getText() ).intValue();
   }
   catch( Exception exc){ tDias.setText("0");datos.dias=0;   }

  try{  datos.sueldo=new Double(tSueldo.getText() ).doubleValue();
  }
  catch( Exception exc){ tSueldo.setText("0");datos.sueldo=0;  }
  grabarDatos(datos);
}

class EventosBoton implements ActionListener{
   public void actionPerformed(ActionEvent evento){
            Object fuente=evento.getSource();
             if(fuente==modificar) {recuperaDatos();}
             if (fuente==buscar) {buscar.setEnabled(false); formaCambios();}
    }
}

class ManejoDeEventosFrame extends WindowAdapter{
        public void windowClosing(WindowEvent evento){
                  if (evento.getSource()==vCaptura){
                       if  (v!=null){ v.setVisible(false); v.dispose();}
                       vCaptura.setVisible(false);
                       vCaptura.dispose();
                  }
                  if (evento.getSource()==v){  v.setVisible(false);
                                               v.dispose();
                                               buscar.setEnabled(true);
                   }
           }
       }
    }
}
class PruebaMysqlCambios{
  public static void main(String args[]) {
            MysqlCambios mysql=new MysqlCambios();
            mysql.conectar();
            mysql.cambios() ;
  }
 }
```

Ventanas del ejemplo 9.10.1

9.11 Eliminación de Registros

El comando para borrar un registro en SQL es DELETE, se envía a MySQL desde Java a través del método executeUpdate de Statement. La sintaxis de DELETE es

DELETE FROM <tabla> where <campo=criterio>;

9.11.1 Ejemplo de Eliminación de Registros

```
import java.awt.*;
import java.sql.*;
import java.awt.event.*;

class Persona{   String nombre;
                 double sueldo;
                 int  dias;
}

class MysqlBajas{   Connection conexion;

        void conectar(){
            try {
                    String driver = "com.mysql.jdbc.Driver";
                    System.out.println( "\n=> loading driver:" );
                    Class.forName(driver).newInstance();
                    String url = "jdbc:mysql://localhost/test";
                    System.out.println( "\n=> connecting:" );
                    conexion=DriverManager.getConnection( url, "root", "ginger" );
                    System.out.println( "OK" );
                  }

                    catch( Exception x ) {       System.err.println( x );
             }
          }

          void bajas(){      Formas forma=new Formas();
                             forma.busqueda();
          }

          void borrarDatos(Persona datos){
               String comando=new String("DELETE FROM datos WHERE nombre='"+ datos.nombre + "';");
               System.out.println(comando);
                try{
                    Statement stmt=conexion.createStatement();
                    stmt.executeUpdate(comando);
                    System.out.println("Registro modificado!");
                }

               catch( java.sql.SQLException er){
                                    System.out.println("No se pudo realizar la operacion");
              }

}
```

continua 9.11.1 Ejemplo de Eliminación de Registros

```
class Formas{
        TextField tNombre,tDias,tSueldo,tBuscar;
        Button borrar;
        Frame v;
        EventosBoton monitorBoton;
        ManejoDeEventosFrame monitorFrame;
        ManejaEventosTexto monitorTexto;
        ResultSet result;

        void busqueda(){
                        v=new Frame("Bajas");
                        v.setLayout(new FlowLayout() );
                        v.add(new Label("Nombre") );
                        v.add(tBuscar= new TextField (20) );
                        v.add(new Label("Dias") );
                        v.add(tDias=new TextField("0",2) );
                        v.add(new Label("Sueldo") );
                        v.add(tSueldo=new TextField("0",10) );
                        v.add(borrar=new Button("borrar"));
                        monitorBoton=new EventosBoton();
                        v.addWindowListener(monitorFrame =new ManejoDeEventosFrame() );
                        monitorTexto=new ManejaEventosTexto();
                        tBuscar.addTextListener(monitorTexto);
                        borrar.addActionListener(monitorBoton);
                        v.pack();
                        v.setVisible(true);
        }


       void muestraDatos(){
                Persona datos=new Persona();
                try{
                    Statement stmt=conexion.createStatement();
                  result=stmt.executeQuery("SELECT * FROM datos WHERE nombre=' " +
                                                                        tBuscar.getText() + "';");
                  if (result.next()) {
                                        datos.nombre=result.getString(1);
                                        datos.dias=result.getInt(2);
                                        datos.sueldo=result.getFloat(3);
                                        tBuscar.setEditable(false);
                                        tDias.setText(new Integer(datos.dias).toString()  );
                                        tDias.setEditable(false);
                                        tSueldo.setText(new Double(datos.sueldo).toString() );
                                        tSueldo.setEditable(false);
                                        borrarDatos(datos);
                                        tBuscar.setEditable(true);
                   }
                }

                catch( java.sql.SQLException er){
                                System.out.println("No se pudo realizar la operacion");
                }

        }
```

continua 9.11.1 Ejemplo de Eliminación de Registros

```
        class EventosBoton implements ActionListener{
                public void actionPerformed(ActionEvent evento){
                        Object fuente=evento.getSource();
                        if(fuente==borrar) {borrar.setEnabled(false); muestraDatos(); }

                        }
                }

        class ManejaEventosTexto implements TextListener{
                public void textValueChanged(TextEvent e){
                Object fuente=e.getSource();
                        if (fuente==tBuscar){  borrar.setEnabled(true);
                                               tDias.setText("");
                                               tSueldo.setText("")
                        }
                }
        }

        class ManejoDeEventosFrame extends WindowAdapter{
                public void windowClosing(WindowEvent evento){
                        if (evento.getSource()==v){   v.setVisible(false);
                                                      v.dispose();
                        }
                }
        }
    }
}


class PruebaMysqlBajas{
  public static void main(String args[]) {
        MysqlBajas mysql=new MysqlBajas();
        mysql.conectar();
        mysql.bajas() ;
  }
}
```

Pantallas producidas por el programa ejemplo de eliminacion de registros