Java DataBase Connectivity

JDBC

Elizabeth León Guzmán, Ph.D.
eleonguz@unal.edu.co

Jonatan Gómez Perdomo, Ph. D.
jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.
aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D. (c)
eccubidesg@unal.edu.co

Carlos Andres Sierra, M.Sc.
casierrav@unal.edu.co

Research Group on Data Mining – Grupo de Investigación en Minería de Datos – (Midas)
Research Group on Artificial Life – Grupo de Investigación en Vida Artificial – (Alife)
Computer and System Department
Engineering School
Universidad Nacional de Colombia

Agenda

Agenda

Java y las BD Relacionales

- Sin ser una regla, las bases de datos relacionales suelen comportarse muy bien con los modelos orientados a objetos.
- Existe una sencilla modelación debido a que la estructura de las tablas (MER) es fácilmente transferible a un diseño de objetos (Diagrama de Clases).
- Pensar en el conjunto de registros de una tabla es equivalente a pensar en una lista de objetos instanciados de una clase.
- Java ha tenido una fuerte tradición de trabajar con bases de datos como MySQL y Oracle (de allí que Oracle hoy en día sea el dueño del lenguaje de programación Java y de MySQL).

Conectores de Bases de Datos

En general, para facilitar la comunicación entre un lenguaje de programación y un gestor de base de datos, se usa un **conector**.

Estos conectores facilitan la comunicación para poder realizar las transacciones desde la base de datos, asegurando la compatibilidad entre los tipos de datos a lugar y el almacenamiento de tablas en listas de elementos (normalmente).

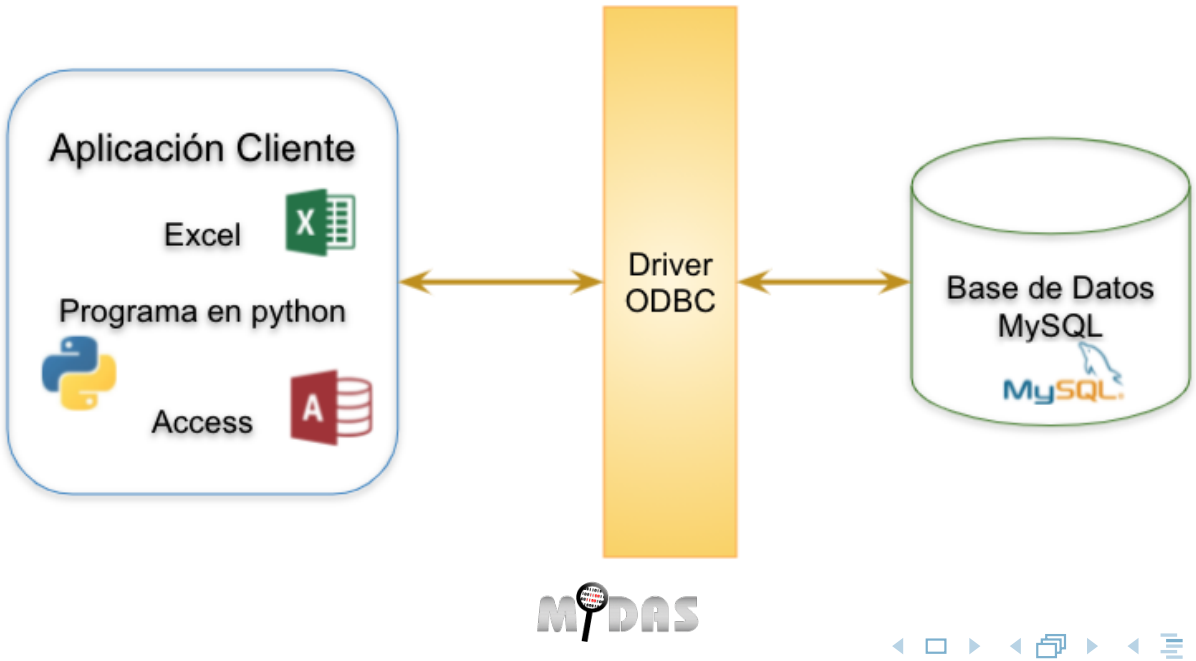
Sirven como intermediador para asegurarse que no se van a tener problemas entre la integración del gestor de base de datos y la lógica desarrollada en el lenguaje de programación.

Agenda

Open DataBase Connectivity - ODBC

Definición

ODBS es un método estándar para compartir datos entre bases de datos y programas (los programas pueden ser otras bases de datos, hojas de cálculo, programas en python, etc.). En esencia, los controladores ODBC utilizan SQL para acceder a la información consignada en una base de datos.

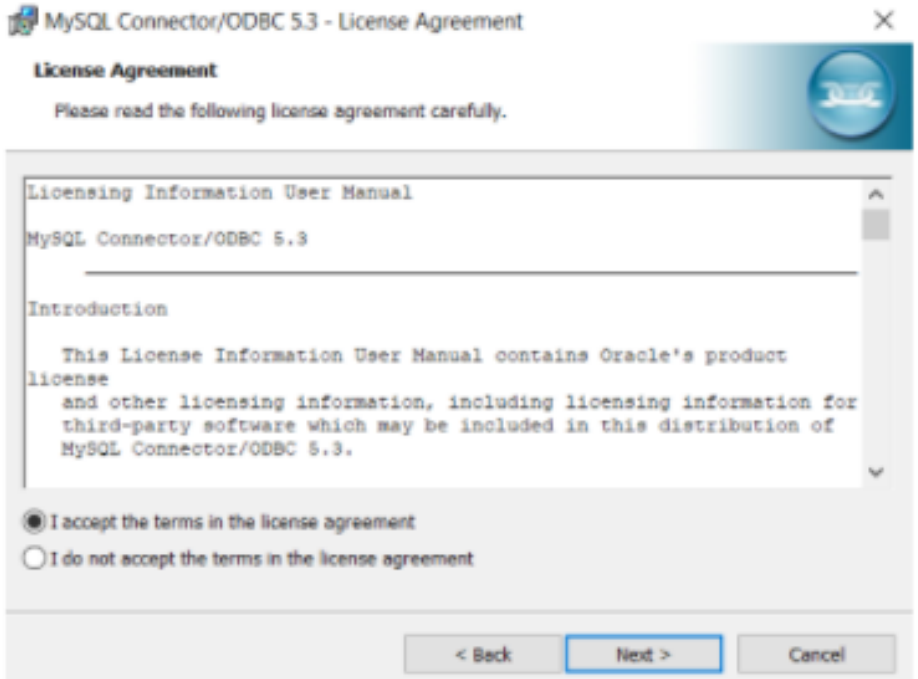


ODBC para MySQL (I)

Para instalar el ODBC de MySQL puede usar este link; se debe seleccionar el *MSI Installer Connector - ODBC*.
Cuando descargue, tendrá una ventana como la siguiente.
Importante que revise las versiones para determinar compatibilidad.

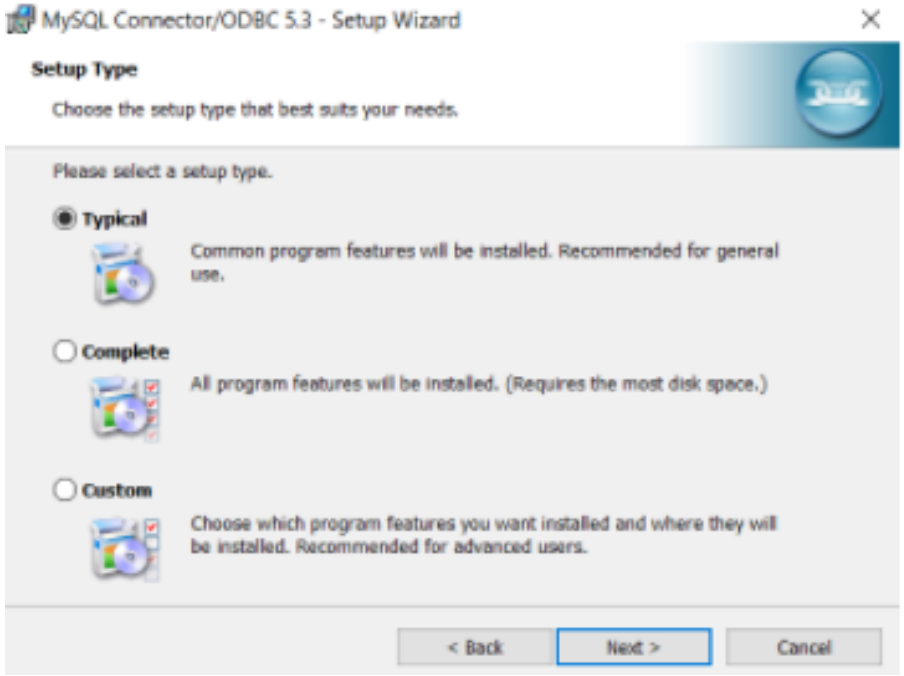
ODBC para MySQL (II)

Toda aplicación siempre mostrará los términos y condiciones. En este caso, de nuevo hay que seleccionar la opción de aceptar y luego continuar.



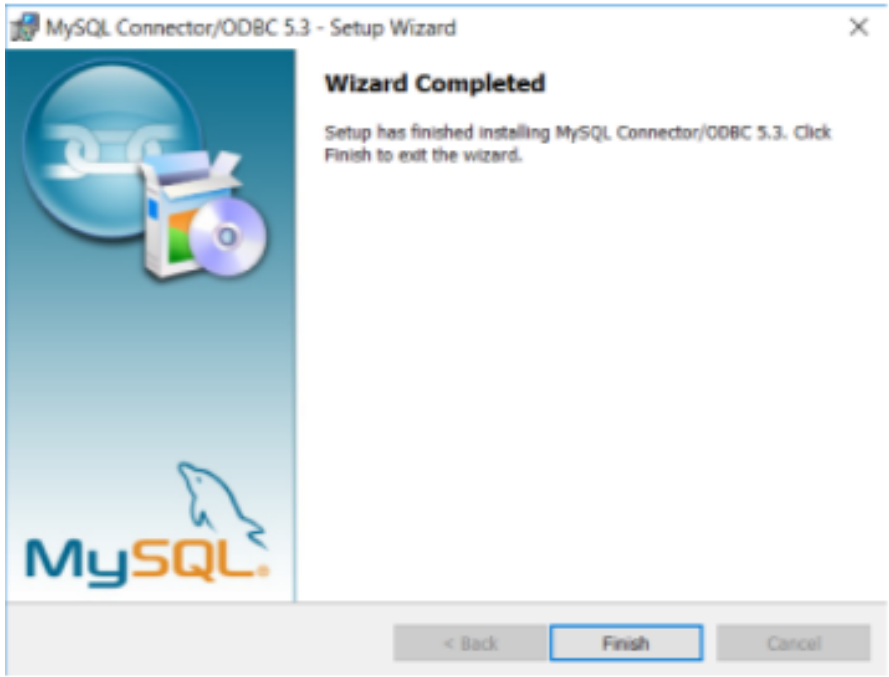
ODBC para MySQL (III)

Aunque hay distintas versiones válidas, cuando se está explorando una nueva herramienta se recomienda siempre instalar la versión básica o típica.



ODBC para MySQL (IV)

Así de simple es instalar el ODBC para MySQL, es poco probable que vaya a tener más errores. Esta sería la ventana que debería obtener al final, completamente instalado.



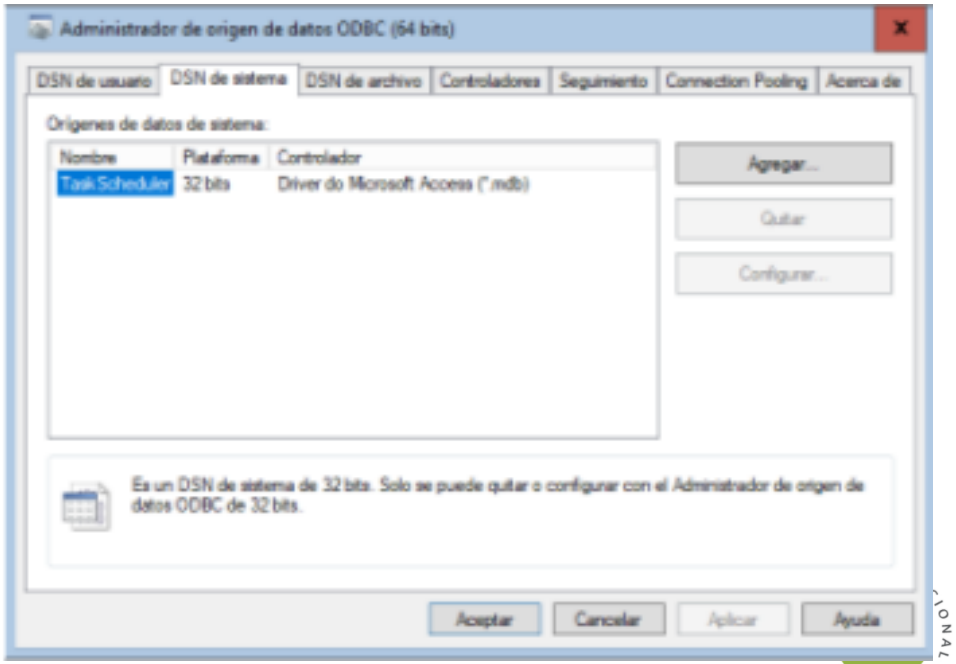
Configurar Origen de Datos (I)

Panel de Control de Windows

Origen de Datos ODBC

Datos e información necesaria para tener acceso a esos datos desde programas y bases de datos que admitan el protocolo ODBC (se debe configurar en el sistema operativo)

En Windows, para agregar un nuevo origen de datos debe hacer:
Panel de control → Herramientas administrativas → Orígenes de datos ODBC.

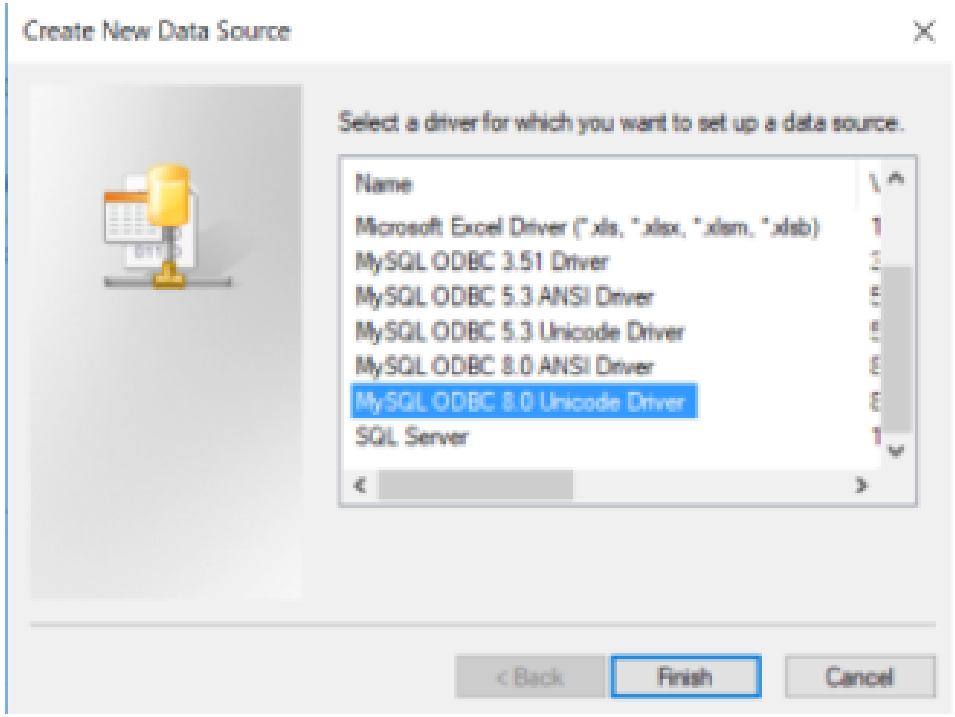


Aparece la siguiente ventana en la cual se debe seleccionar la pestaña de *DSN de sistema*.

Configurar Origen de Datos (II)

Panel de Control de Windows

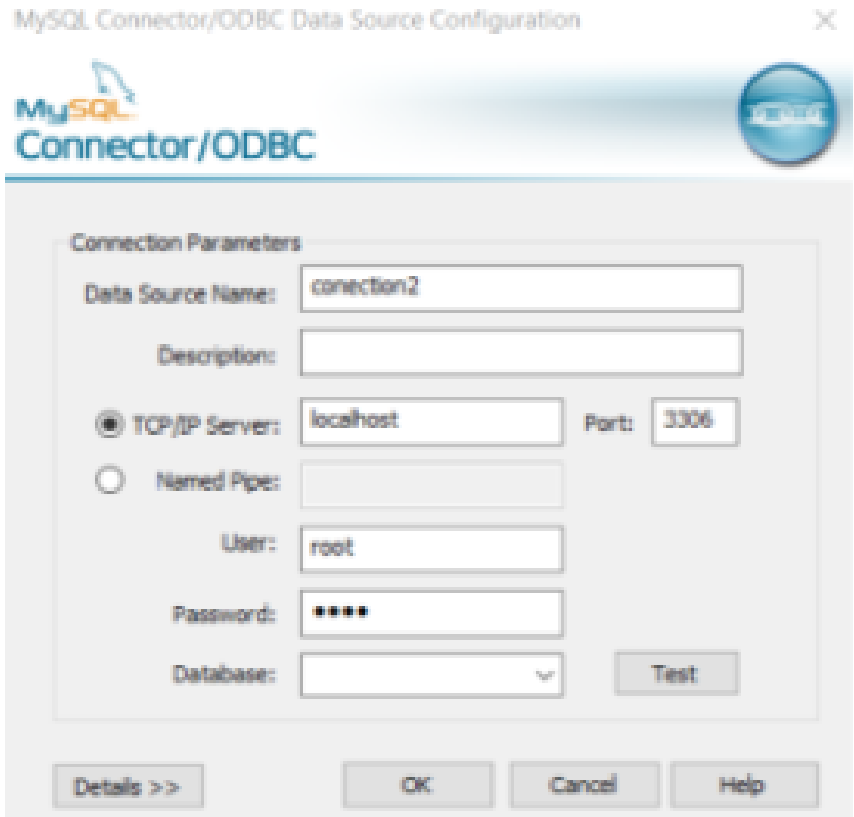
Se debe hacer clic en *Agregar*, va a aparecer la siguiente ventana, en la cual se elige el controlador respectivo; en este caso es *MySQL ODBC 8.0 Unicode Drive*.



Configurar Origen de Datos (III)

Aparecen los siguientes campos de configuración:

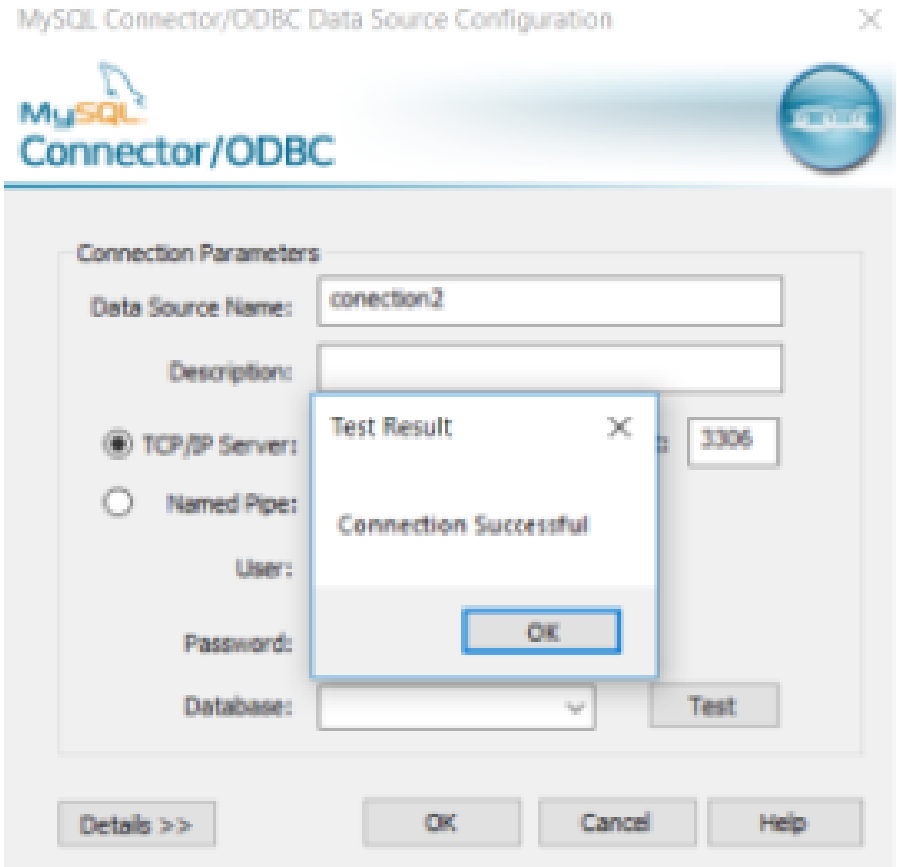
- Data Source Name: Nombre con el que identificar en Windows la Base de Datos MySQL a la cual se va a acceder.
- Server: Aquí se indica la IP del servidor o su nombre. (Ej 127.0.0.1)
- User: El usuario con privilegios para acceder a la Base de Datos.
- Password: El password para acceder de ese usuario.
- Database: Nombre de la Base de datos que va a ser accedida.
- Port: Es el puerto por donde se conecta a la Base de datos, por defecto usará el 3306.



Configurar Origen de Datos (IV)

Si se hace clic en *Test*, se puede ver si la conexión ha sido exitosa o hay algún error de conexión.

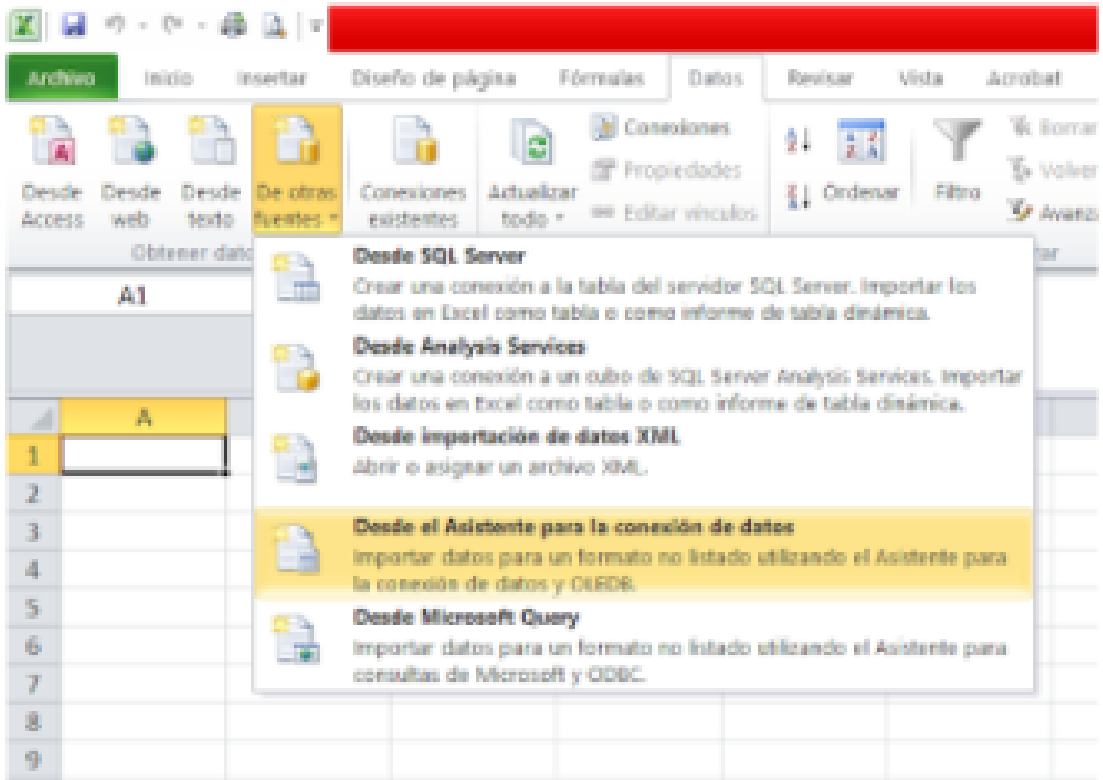
Esto es importante realizarlo para verificar que haya sido correctamente creado el origen de datos, y que no existan algunos problemas de credenciales o error en nombres.



ODBC con Excel (I)

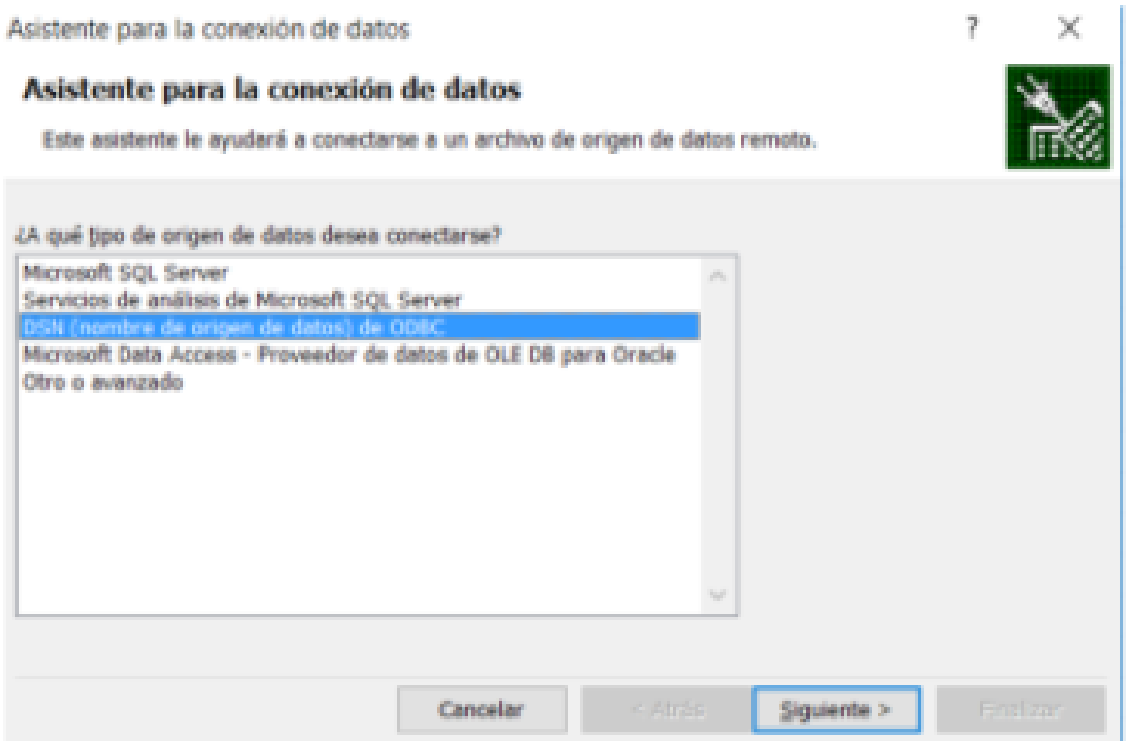
Se realizará un ejemplo usando ODBC para conexión a MySQL desde excel.

Se crea un nuevo libro en Excel, y en: Datos → De otras fuentes → Desde el Asistente para la conexión de Datos.



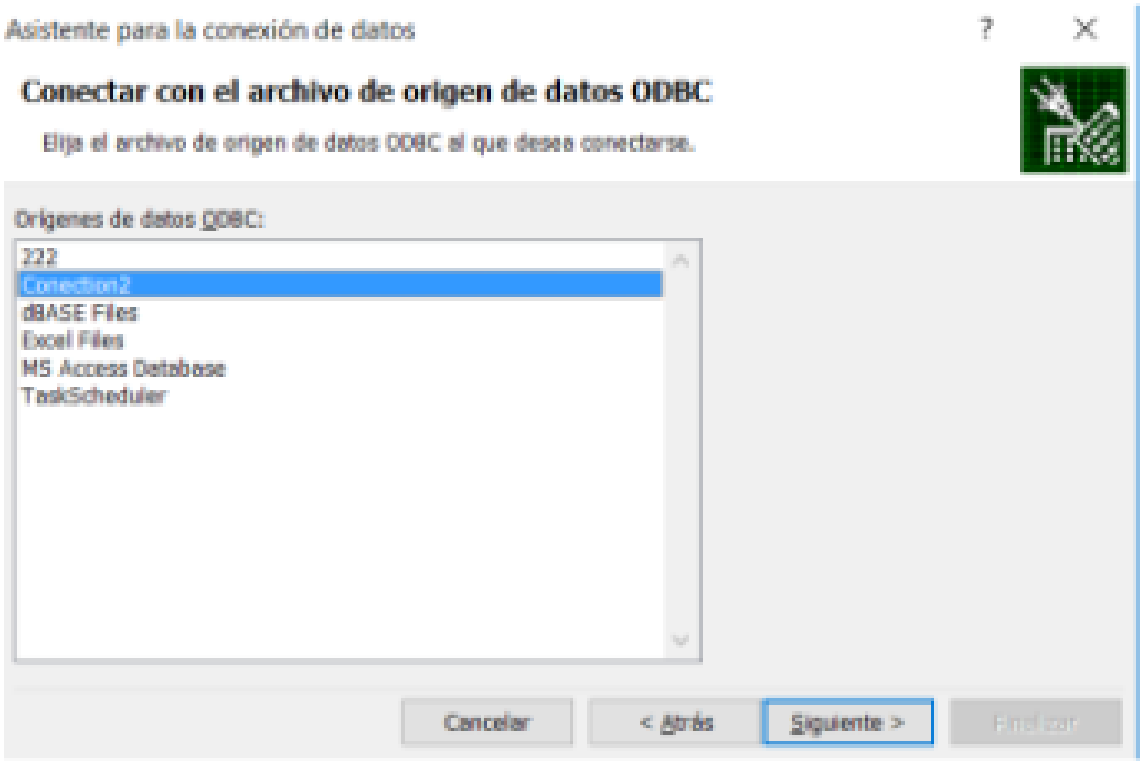
ODBC con Excel (II)

Aparece una ventana para seleccionar: archivos de origen de datos



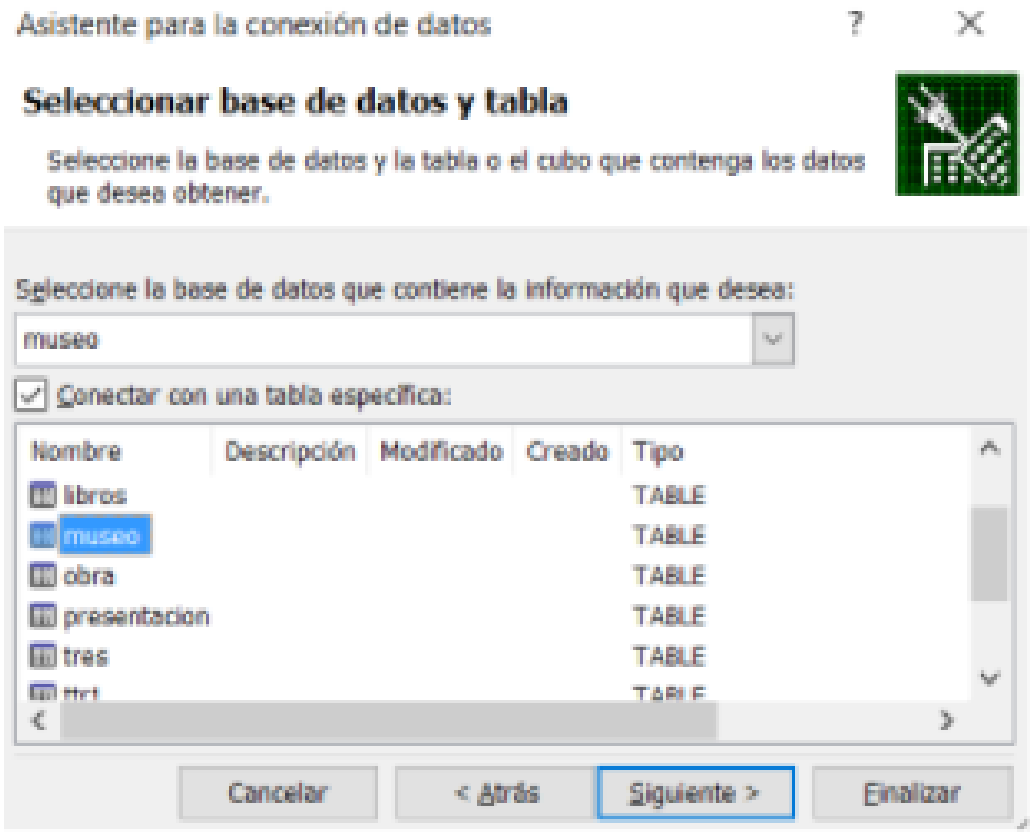
ODBC con Excel (III)

Se selecciona *DSN (nombre de origen de datos) de ODBC*, se presiona *siguiente* para elegir el archivo de **Origen de Datos ODBC** al que se desea conectar, que fue el ya creado, en este caso *Conection2*.



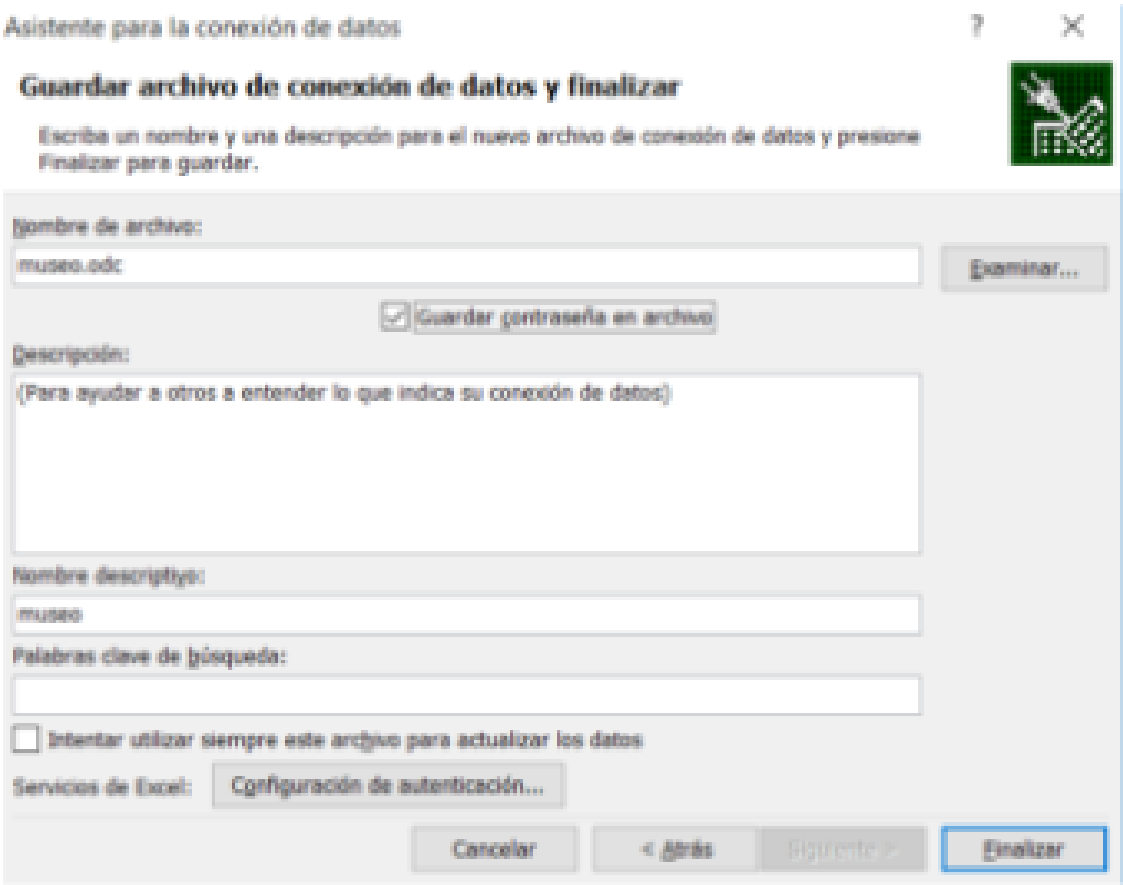
ODBC con Excel (IV)

Se puede seleccionar la base de datos que contiene la información que se desea.También se encuentra la opción de conectar alguna tabla específica, donde se seleccionan las tablas que se necesiten.



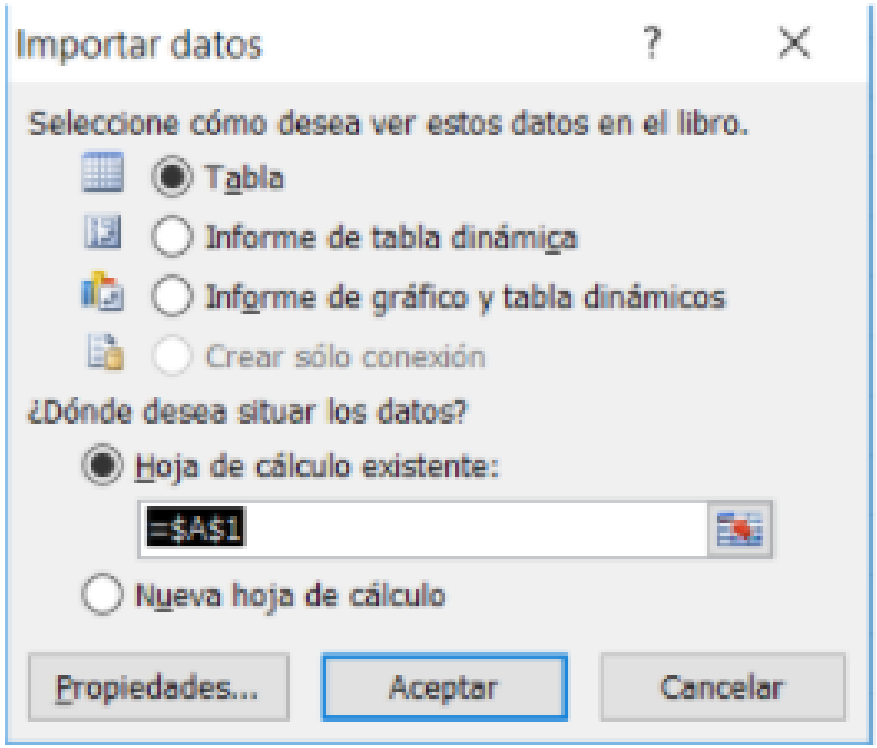
ODBC con Excel (V)

En la siguiente ventana, se recomienda Guardar la Contraseña en archivo seleccionando el *Check Box* correspondiente.



ODBC con Excel (VI)

Luego, se puede elegir la ubicación dentro de la hoja de cálculo para colocar los datos obtenidos desde el origen.



ODBC con Excel (VII)

Así, se verá la tabla cargada en una hoja de cálculo de Excel:

	A	B	C
1	Id_museo	Nom_museo	
2	101	Louvre	
3	102	Met	
4	205	Shangai	
5	304	Britanico	
6			
7			
8			
9			

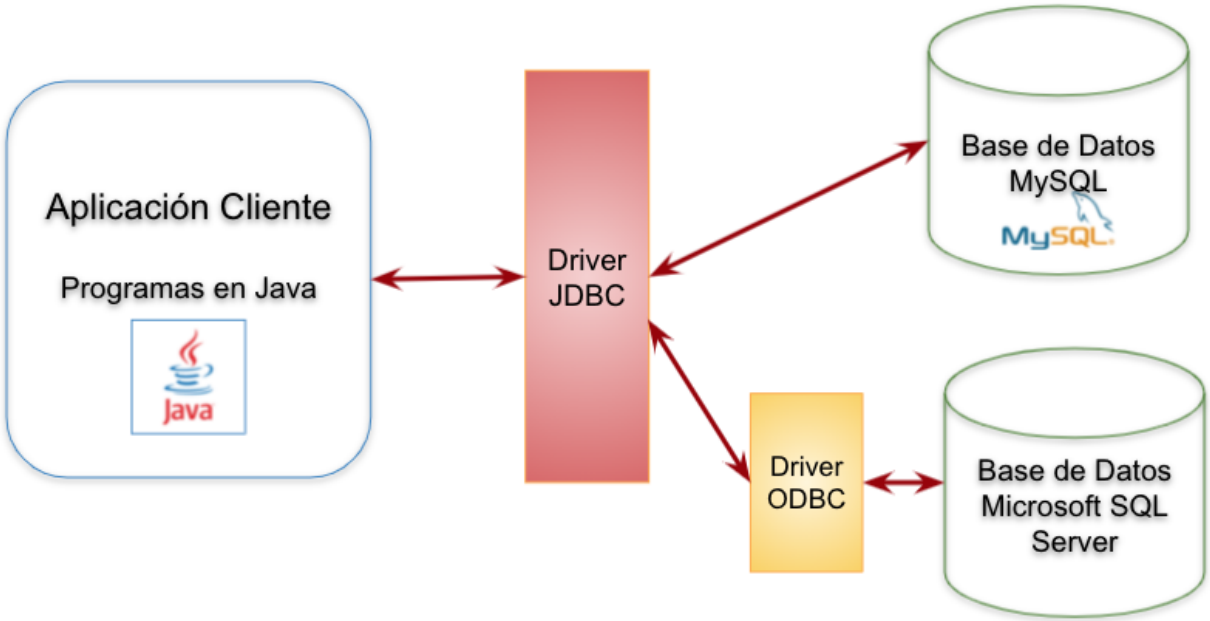
Agenda

Agenda

Java DataBase Connectivity - JDBC (I)

JDBC (Java DataBase Connectivity) es un API que brinda un conjunto de interfaces y clases para acceder desde Java a cualquier motor de base de datos que lo implemente.

JDBC se abstrae de los detalles específicos del motor, permitiendo así conectarse prácticamente de la misma manera a cualquier base de datos.



Java DataBase Connectivity - JDBC (II)

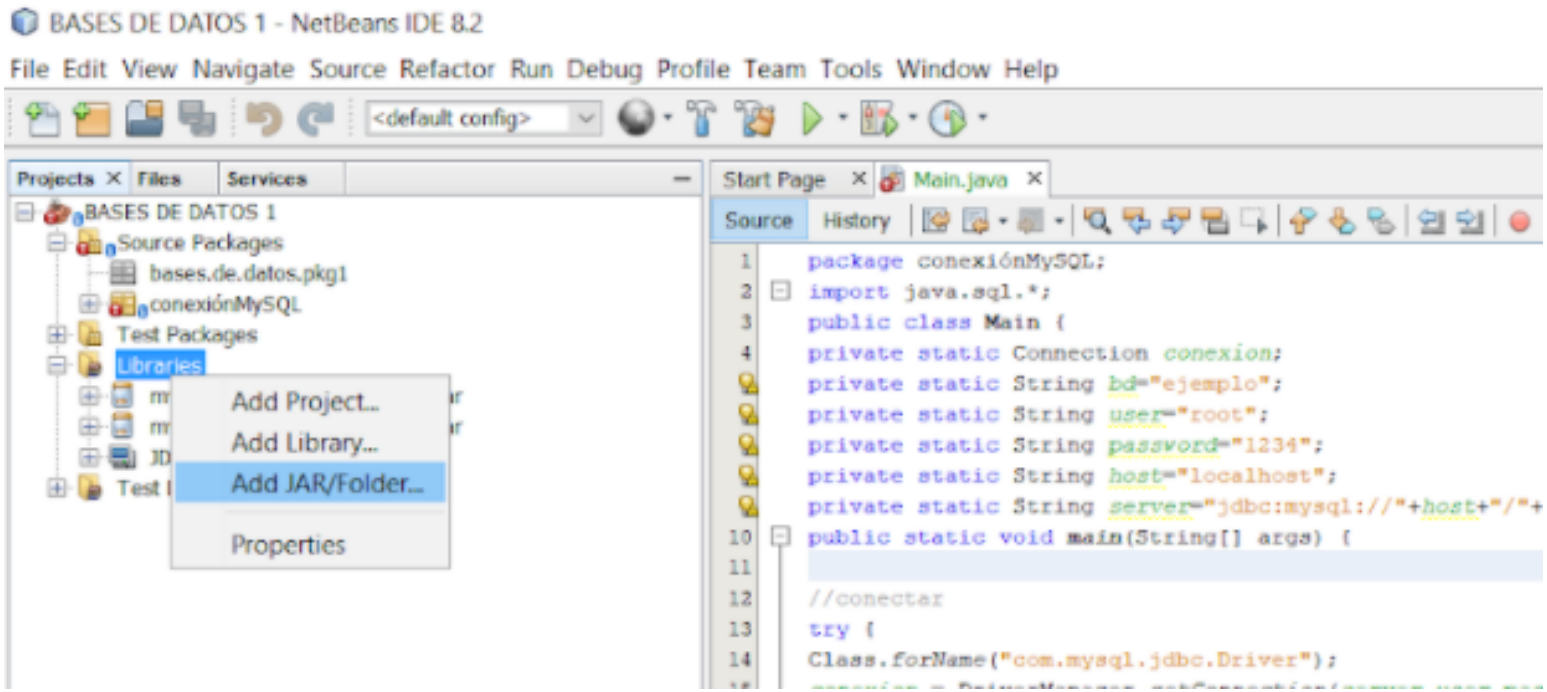
Para conectar Java con cualquier base de datos lo primero es conseguir el **driver** adecuado. Un **Driver** es una clase que sirve como interfaz entre nuestra aplicación y la base de datos.

Driver de MySQL

Es lógico que Java (JDK) no cuenta con todos los Drivers para poderse conectar con todos los motores de base de datos existentes en el mercado, por lo que en algunos casos es necesario descargarlos de internet. En el caso de MySQL se puede descargar desde:
http://dev.mysql.com/downloads/connector/j/5.1.html

Java DataBase Connectivity - JDBC (III)

Para NetBeans, en el árbol de proyecto se da click con el botón secundario del ratón sobre “Bibliotecas” para sacar el menú y elegimos “Agregar JAR/Folder”, y se busca el drive de MySQL. Esto se hará para cada proyecto que se esté trabajando.



Propiedades Comunes

Hay unas propiedades comunes que siempre necesitan ser definidas cuando se está realizando una conexión por JDBC. Estas se muestran a continuación:

- URL de conexión: Una cadena de texto que el driver JDBC usa para conectarse a la base de datos. Normalmente tiene información como la ubicación de la base de datos, nombre de la base de datos, y cualquier otra propiedad de configuración.
  Ejemplo: *jdbc:mysql://localhost:3306/museo?serverTimezone=UTC*
- Driver: Esl nombre definido para la clase usada por el driver.En este caso, usamos el driver de MySQL: *com.mysql.cj.jdbc.Driver*
- Username & password: Credenciales de acceso de la cuenta de MySQL.

Agenda

Conexión con JDBC

Ejemplo

```
String dbURL =
        "jdbc:mysql://localhost:3306/libreria";
String username = "root";
String password = "secret";
//conectar
try {
    Connection conn = DriverManager.getConnection(
                            dbURL,username,password);

     if (conn != null) {
         System.out.println("Conectado");
     }
} catch (SQLException ex) {
     ex.printStackTrace();
}
```

Recomendaciones de Uso

Se recomienda usar JDBC solamente cuando se vayan a hacer transacciones sencillas y en baja cantidad. Para aplicaciones donde se deben definir muchas transacciones se recomiendan otras maneras de hacer las conexiones.

Se debe tener el driver correcto para el gestor de base de datos a utilizar, de no ser así, la aplicación no va a funcionar. Esto quiere decir que hay una alta dependencia del driver a usar.

Agenda

CRUD usando JDBC

Las operaciones basicas en bases de datos: **Crear**, **Consultar**, **Actualizar** y **Borrar** por sus siglas en ingles CRUD (Create, Retrieve, Update, Delete) encuentran su equivalente en el lenguaje SQL como INSERT, SELECT, UPDATE y DELETE.

A continuación se verá cómo realizar estas operaciones usando JBDC con una base de datos MySQL. Sin embargo, esto se puede aplicar también a otros sistemas de base de datos, ya que la sintaxis de consulta utilizada es SQL estándar, que es compatible con todos los sistemas de bases de datos relacionales.

CRUD usando JDBC - Create

Ejemplo

```
String sql = "INSERT INTO libro(libId,libNombre,
libPub,ediId,autId,libPrecio) VALUES( VALUES (?,
?, ?, ?, ?, ?)";
PreparedStatement statement =
                    conn.prepareStatement(sql)

statement.setInt(1, 1010);
statement.setString(2, "La Hojarasca");
statement.setInt(3, 1955);
statement.setInt(4, 1);
statement.setInt(5, 1);
statement.setDouble(6, 95000.0);
int rowsInserted = statement.executeUpdate();
if (rowsInserted > 0) {
    System.out.println("Inserción exitosa!");
}
```

CRUD usando JDBC - Retrieve

Ejemplo

```
String sql = "SELECT * FROM libro";
Statement statement = conn.createStatement();
ResultSet result = statement.executeQuery(sql);

int count = 0;
while (result.next()){
  String titulo = result.getString(2);
  Int pub = result.getInt(3)
  Double costo = result.getDouble(6);

  System.out.println("Título: " +
        titulo + "Año publicación: "  + pub +
        " Costo: " + costo);
}
```

CRUD usando JDBC - Update

Ejemplo

```
String sql = "UPDATE libro SET libNombre=?,
    libPub=?, libPrecio=? WHERE libId=?";

PreparedStatement statement =
                    conn.prepareStatement(sql);
statement.setString(1, "Crónica de una muerte
                        anunciada");
statement.setInt(2, 1981);
statement.setDouble(3, 100000.0);
statement.setInt(4, 1010);

int rowsUpdated = statement.executeUpdate();
if (rowsUpdated > 0) {
    System.out.println("El registro fue " +
                "actualizado exitosamente!");
}
```

CRUD usando JDBC - Delete

Ejemplo

```
String sql = "DELETE FROM libro WHERE libId=?";

PreparedStatement statement =
                        conn.prepareStatement(sql);
statement.setInt(1, 1010);

int rowsDeleted = statement.executeUpdate();
if (rowsDeleted > 0) {
    System.out.println("Borrado exitoso!");
}
```