

jose 7:04 pm el March 18, 2018

Etiquetas: acceso a datos ( 13 ), java ( 20 )

## Conexión a distintas BD con JDBC

a través de [Acceso a Datos](#)

En Windows supongamos que la base de datos «**ejemplo**» la tenemos en las carpetas D:\DB\SQLite, D:\DB\HSQLDB\ejemplo, D:\DB\Hz y D:\DB\DERBY.

En Linux en las carpetas /home/usuario/DB/SQLITE, /home/usuario/DB/ HSQLDB, /home/usuario/DB/ Hz y /home/usuario/DB/ DERBY/.

**Para realizar las pruebas necesitaremos tener el conector Java (archivo JAR) correspondiente para cada una de las bases de datos.**

### Conexión a SQLite.

Para conectarnos a SQLite necesitamos la librería sqlite-jdbc-3.8.11.2.jar que se puede descargar desde la URL <https://bitbucket.org/xerial/sqlite-jdbc/downloads>. Partimos del programa Java inicial que recorre la tabla departamentos (su nombre es Main.java) de la base de datos ejemplo de MySQL. En el entorno gráfico que usemos para ejecutar el programa incluimos el fichero JAR o lo incluimos en el CLASSPATH si lo ejecutamos desde la línea de comandos.

En el programa Java cambiamos dos cosas: la carga del driver, en este caso se llama org.sqlite.JDBC y la conexión a la base de datos:

```
Class.forName("org.sqlite.JDBC");  
Connection conexion = DriverManager.getConnection("jdbc:sqlite:D:/DB/SQLITE/ejemplo.db");
```

El ejemplo en Linux es similar, solo habría que cambiar en la conexión la carpeta donde se encuentra la base de datos: "jdbc:sqlite:/home/usuario/DB/SQLITE/ejemplo.db".

### Conexión a Apache Derby.

Para conectarnos a Apache Derby necesitamos la librería **derby.jar** (que se encuentra en la carpeta donde se instaló Derby: (db-derby-10.12.1.1-bin/lib).

El driver para la conexión a la base de datos se llama **org.apache.derby.jdbc.EmbeddedDriver**:

```
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");  
Connection conexion =  
DriverManager.getConnection ("jdbc:derby:D:/DB/DERBY/ejemplo");
```

### **Conexión a HSQLDB.**

Para conectarnos a HSQLDB necesitamos la librería `hsqldb.jar` que se puede obtener de la carpeta `lib` obtenida al descomprimir el fichero `hsqldb-2.3.3.zip`. En este caso el driver se llama `org.hsqldb.jdbcDriver` y la conexión a la base de datos es la siguiente:

```
Class.forName("org.hsqldb.jdbcDriver");  
Connection conexion =  
DriverManager.getConnection ("jdbc:hsqldb:file:D:/DB/HSQLDB/ejemplo/ejemplo");
```

### **Conexión a H2.**

Para conectarnos a H2 necesitamos la librería `h2-1.4.191.jar` que se puede obtener de la carpeta `bin` en la que se encuentra al descomprimir el fichero `h2-2016-01-21.zip`. El driver se llama `org.h2.Driver` y la conexión a la base de datos es la siguiente:

```
Class.forName(«org.h2.Driver»);  
Connection conexion =  
DriverManager.getConnection («jdbc:h2:D:/DB/H2/ejemplo/ejemplo», «sa»,»);
```

En este caso es necesario incluir el nombre del usuario y la clave en la conexión. El nombre es «sa» y la clave se dejó en blanco cuando se creó la base de datos.

### **Conexión a Access.**

Para conectarnos a una base de datos Access necesitamos las siguientes librerías: `commons-lang-2.6.jar`, `commons-logging-1.1.1.jar`, `hsqldb.jar`, `jackcess-2.1.2.jar`, `ucanaccess-3.0.2.jar`, y `ucanload.jar`. Se pueden descargar de la URL: <http://ucanaccess.sourceforge.net/site.html>. Al acceder al sitio podremos descargar un fichero similar a `UCanAccess-3.0.4-src.zip` con ejemplos. O el fichero `UCanAcces-3.0.4-bin.zip` que contiene los JAR.

Para conectarnos a una base de datos Access escribiremos:

```
Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");  
Connection conn =  
DriverManager.getConnection ("jdbc:ucanaccess://mibasedatosaccess");
```

Por ejemplo, si quiero conectarme a la base de datos ejemplo.accdb, que tengo guardada en la carpeta raíz del proyecto, en la conexión escribiré lo siguiente;

```
Connection conn = DriverManager.getConnection ("jdbc:ucanaccess://./ejemplo.accdb");
```

### **Conexión a MySQL.**

Para conectarnos a MySQL necesitamos la librería mysql-connector-java-5.1.38-bin.jar, que podemos descargar desde la URL <http://dev.mysql.com/downloads/connector/j/>. Se descarga un fichero ZIP, y dentro de él se encuentra el JAR. El driver se llama com.mysql.jdbc y la conexión es la siguiente:

```
Class.forName("com.mysql.jdbc.Driver");  
Connection conexion =  
DriverManager.getConnection ("jdbc:mysql://localhost/ejemplo", "ejemplo", "ejemplo");
```

### **Conexión a Oracle.**

Para conectarnos mediante JDBC usamos el driver JDBC Thin. Se puede descargar desde la página web de Oracle . Necesitamos saber el nombre de servicio que usa la base de datos para incluirlo en la URL de la conexión. Normalmente para la versión Express Edition el nombre es XE. El driver se llama oracle.jdbc.driver.OracleDriver, y la conexión a la base de datos es la siguiente:

```
Class.forName("oracle.jdbc.driver.OracleDriver");  
Connection conexion =  
DriverManager.getConnection ("jdbc:mysql:thin:@localhost:1521:XE", "ejemplo", "ejemplo");
```

### **Ejecución de sentencias de descripción de datos.**

La interfaz DatabaseMetaData proporciona información sobre la base de datos a través de múltiples métodos de los cuales es posible obtener gran cantidad de información. Muchos de estos métodos devuelven un ResultSet:

#### **getTables()**

Proporciona información sobre las tablas y vistas de la base de datos.

#### **getColumns()**

Devuelve información sobre las columnas de una tabla.

#### **getPrimaryKeys()**

Proporciona información sobre las columnas que forman la clave primaria de una tabla.

#### **getExportedKeys()**

Devuelve información sobre las claves ajenas que utilizan la clave primaria de una tabla.

#### **getImportedKeys()**

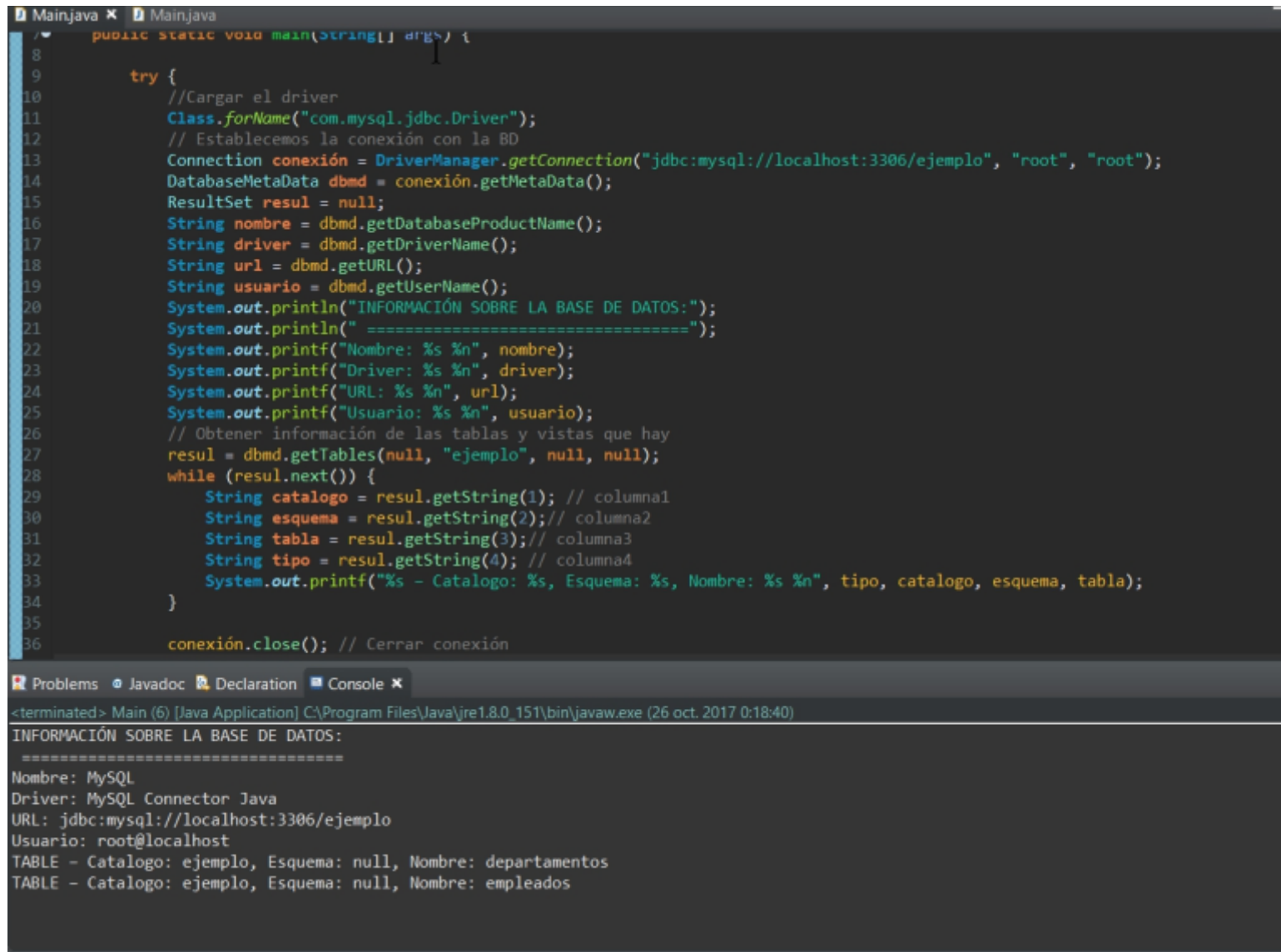
Devuelve información sobre las claves ajenas existentes en una tabla.

#### **getProcedures()**

Devuelve información sobre los procedimientos almacenados.

el método **getMetaData()** de la interfaz Connection devuelve un objeto DataBaseMetaData que contiene información sobre la base de datos representada por el objeto Connection:

[+ ampliar código fuente](#)



```

Main.java x Main.java
public static void main(String[] args) {
    try {
        //Cargar el driver
        Class.forName("com.mysql.jdbc.Driver");
        // Establecemos la conexión con la BD
        Connection conexión = DriverManager.getConnection("jdbc:mysql://localhost:3306/ejemplo", "root", "root");
        DatabaseMetaData dbmd = conexión.getMetaData();
        ResultSet resul = null;
        String nombre = dbmd.getDatabaseProductName();
        String driver = dbmd.getDriverName();
        String url = dbmd.getURL();
        String usuario = dbmd.getUserName();
        System.out.println("INFORMACIÓN SOBRE LA BASE DE DATOS:");
        System.out.println("=====");
        System.out.printf("Nombre: %s %n", nombre);
        System.out.printf("Driver: %s %n", driver);
        System.out.printf("URL: %s %n", url);
        System.out.printf("Usuario: %s %n", usuario);
        // Obtener información de las tablas y vistas que hay
        resul = dbmd.getTables(null, "ejemplo", null, null);
        while (resul.next()) {
            String catalogo = resul.getString(1); // columna1
            String esquema = resul.getString(2); // columna2
            String tabla = resul.getString(3); // columna3
            String tipo = resul.getString(4); // columna4
            System.out.printf("%s - Catalogo: %s, Esquema: %s, Nombre: %s %n", tipo, catalogo, esquema, tabla);
        }
        conexión.close(); // Cerrar conexión
    }
}

```

Problems Javadoc Declaration Console x

```

<terminated> Main (6) [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (26 oct. 2017 0:18:40)
INFORMACIÓN SOBRE LA BASE DE DATOS:
=====
Nombre: MySQL
Driver: MySQL Connector Java
URL: jdbc:mysql://localhost:3306/ejemplo
Usuario: root@localhost
TABLE - Catalogo: ejemplo, Esquema: null, Nombre: departamentos
TABLE - Catalogo: ejemplo, Esquema: null, Nombre: empleados

```

**getTables()** devuelve un objeto ResultSet que proporciona información sobre las tablas y vistas de la base de datos.

**getColumns()** devuelve un objeto ResultSet con información sobre las columnas de una tabla o tablas

Por ejemplo, **getColumns(null, "ejemplo", "departamentos", "d%")** obtiene todos los nombres de columna que empiezan por la letra de en la tabla departamentos y en el esquema de nombre ejemplo. El valor null en los 4 parámetros indica que obtiene información de todas las columnas y tablas del esquema actual.

Ejemplo:

Mostrar informacion de todas las columnas de la tabla departamentos:

[+ ampliar código fuente](#)

**getPrimaryKeys()** devuelve la lista de columnas que forman la clave primaria de la tabla especificada.

**Ejemplo:**

**Muestra la clave primaria de la tabla departamentos(con MYSQL).**

```
63
64     System.out.println ("CLAVE PRIMARIA TABLA DEPARTAMENTOS:");
65     System.out.println("=====");
66     ResultSet columnas3 = null;
67
68     columnas3 = dbmd.getPrimaryKeys(null, "ejemplo", "departamentos");
69
70     while (columnas3.next()) {
71         String clavePrimaria = columnas3.getString(4);
72
73         System.out.println("Clave Primaria de la tabla departamentos: "+clavePrimaria);
74
75     }
76
77
78
79
80     conexión.close(); // Cerrar conexión
81
82     } catch (ClassNotFoundException cn) {
83         cn.printStackTrace();
84     } catch (SQLException e) {
85         e.printStackTrace();
86     }
87 }
```

Problems Javadoc Declaration Console

<terminated> Main (6) [Java Application] C:\Program Files\Java\jre1.8.0\_151\bin\javaw.exe (26 oct. 2017 12:43:35)

CLAVE PRIMARIA TABLA DEPARTAMENTOS:

=====

Clave Primaria de la tabla departamentos: dept\_no

**getExportedKeys()** devuelve la lista de todas las claves ajenas que utilizan la clave primaria de la tabla especificada.

**Ejemplo:**

**Muestra las tablas y las claves ajenas que referencian a la tabla departamentos (en este caso solo la tabla empleados)**

```
77
78     System.out.println ("CLAVE AJENA QUE REFERENCIA A TABLA DEPARTAMENTOS:");
79     System.out.println("=====");
80     ResultSet columnas4 = null;
81
82     columnas4 = dbmd.getExportedKeys(null, "ejemplo", "departamentos");
83
84     while (columnas4.next()) {
85         String tablas = columnas4.getString(7);
86         String claveAjena = columnas4.getString(4);
87         System.out.println("Tablas que referencian a la tabla departamentos: "+tablas);
88         System.out.println("Clave Ajena que referencia a la tabla departamentos: "+claveAjena);
89     }
90
91
92
93
94
--
```

Problems Javadoc Declaration Console x

<terminated> Main (6) [Java Application] C:\Program Files\Java\jre1.8.0\_151\bin\javaw.exe (26 oct. 2017 13:04:55)

CLAVE AJENA QUE REFERENCIA A TABLA DEPARTAMENTOS:  
=====

Tablas que referencian a la tabla departamentos: empleados  
Clave Ajena que referencia a la tabla departamentos: dept\_no

**getProcedures()** devuelve la lista de procedimientos almacenados.

Para probar el método `getProcedures()` creamos algunos procedimientos y funciones. Por ejemplo, creamos la función de nombre SUMAR que recibe dos números y devuelve la suma:

#### **Nota: Ejemplo de función con Oracle:**

```
CREATE OR REPLACE FUNCTION SUMAR (N1 NUMBER, N2 NUMBER)
RETURN NUMBER AS
BEGIN
RETURN N1+N2
END SUMAR;
//Para probarla escribimos: SELECT SUMAR (2,22) FROM DUAL;
```

#### **Nota: Ejemplo de función en MySQL:**



```
CREATE FUNCTION SUMAR (N1 INT, N2 INT)  
RETURN INT BEGIN RETURN N1+N2  
END;  
//Para probarla escribimos: SELECT SUMAR (2,22)
```

**Ejemplo:**

**Crea un procedimiento de nombre SUBIDA que sube 100 euros al salario de los empleados del departamento 30. (en mysql).**

```
Query 1 x
1 DELIMITER //
2 • CREATE PROCEDURE SUBIDA(cantSubida INT, dept INT)
3 BEGIN
4 IF dept = 30 THEN
5 UPDATE empleados
6 SET salario = salario + cantSubida WHERE dept_no = dept;
7 END IF;
8 END
9 //
10
```

Query 1 x

```
1 • SELECT * FROM empleados;
2
```

<

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	emp no	apellido	oficio	dir	fecha alt	salario	comision	dept no
	7369	Sanchez	Empleado	7902	1990-12-17	1040.00	HULL	20
	7499	Arroyo	Vendedor	7698	1990-02-20	1500.00	390.00	30
▶	7521	Sala	Vendedor	7698	1991-04-22	1625.00	650.00	30
	7546	Barba	Empleado	7432	2001-02-24	1200.00	230.00	10
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL



Query 1 x

Limit to 1000 r

```
1 • CALL SUBIDA(100,30);
2
3 • SELECT * FROM empleados;
4
```

<

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: I A

	emp no	apellido	oficio	dir	fecha alt	salario	comision	dept no
	7369	Sanchez	Empleado	7902	1990-12-17	1040.00	NULL	20
	7499	Arroyo	Vendedor	7698	1990-02-20	1600.00	390.00	30
▶	7521	Sala	Vendedor	7698	1991-04-22	1725.00	650.00	30
	7546	Barba	Empleado	7432	2001-02-24	1200.00	230.00	10
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
Main.java x
import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.sql.Statement;
8
9 public class Main {
10     public static void main(String[] args) {
11         try{
12             //Cargar el driver
13             Class.forName("com.mysql.jdbc.Driver");
14             //Establecemos la conexión con la BD
15             Connection conexión = DriverManager.getConnection("jdbc:mysql://localhost:3306/ejemplo", "root", "root");
16             // Preparamos la consulta
17             Statement sentencia = conexión.createStatement();
18             String sqlProcedimiento = "CALL SUBIDA(100,30)";
19             ResultSet result = sentencia.executeQuery(sqlProcedimiento);
20             String sql = "SELECT apellido,oficio,salario FROM empleados";
21             ResultSet result1 = sentencia.executeQuery(sql);
22             //Recorremos el resultado para visualizar cada fila
23             while (result1.next()){
24                 System.out.printf("%s, %s, %d %n",
25                     result1.getString(1),
26                     result1.getString(2),
27                     result1.getInt(3));
28             }
29             result1.close(); //Cerrar ResultSet
30             sentencia.close(); //Cerrar Statement
31             conexión.close(); //Cerrar conexión
32         } catch (ClassNotFoundException cn) {
33             System.out.println(cn.getMessage());
34         } catch (SQLException e) {
35             System.out.println(e.getErrorCode());
36         }
37     } //fin de main
38 } //fin de clase

Problems Javadoc Declaration Console x
<terminated> Main (7) [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (27 oct. 2017 18:55:22)
Sanchez, Empleado, 1040
Arroyo, Vendedor, 1600
Sala, Vendedor, 1725
Barba, Empleado, 1200
```

### ResultSetMetaData.

Se pueden obtener metadatos (datos sobre los datos) a partir de un objeto ResultSet mediante la interfaz ResultSetMetaData; es decir podemos obtener más información sobre los tipos y propiedades de las columnas de los objetos ResultSet, como por ejemplo, el número de columnas devueltas, el tipo, el nombre, etc.

[+ ampliar código fuente](#)**Métodos usados:****int getColumnCount()**

Devuelve el número de columnas devueltas por la consulta.

**String getColumnName (int indiceColumna)**

Devuelve el nombre de la columna cuya posición se indica en índiceColumna.

**String getColumnTypeNames(int indiceColumna)**

Devuelve el nombre del tipo de dato específico del sistema de bases de datos que contiene la columna indicada en índiceColumna.

**int isNullable(int indiceColumna)**

Devuelve información sobre las claves ajenas que utilizan la clave primaria de una tabla.

**int getColumnDisplaySize(int indiceColumna)**

Devuelve el máximo ancho en caracteres de la columna indicada en índiceColumna

Anuncios

PRIVACIDAD

INFORMA SOBRE ESTE ANUNCIO

