



# APUNTE Nº4: Introducción al uso de archivos y BBDD con LDP Java

Prof. Dr. Samuel Sepúlveda
DCI-CEIS-UFRO

2016

**CONTENIDOS**

1. Introducción al uso de archivos
2. Introducción al uso de BBDD

1. **INTRODUCCIÓN AL USO DE ARCHIVOS**

Con Java es posible acceder a un archivo al igual que el resto de los demás LDP.

Es posible acceder a archivos del tipo:

- binario
- secuencial
- acceso aleatorio

### 1.1 Uso de archivos

Es necesario considerar una secuencia de tres pasos básicos.

- se abre el archivo (ruta, modo de apertura)
- se procesa el contenido
- se cierra el archivo

¿Cómo realizo esto con Java?

### 1.2 Uso de archivos de acceso aleatorio con Java

Para implementar la secuencia de pasos anterior, es necesario hacer uso de la clase **RandomAccessFile**.

i. Para abrir el archivo: se instancia un objeto para manipular el archivo:

```
RandomAccessFile arch = new RandomAccessFile("archivo.txt","rw");
```

ii. Para leer el contenido del archivo, línea a línea:

```
while ( ( cadena = arch.readLine( ) ) !=null ) {
    System.out.println( cadena );
}
```

iii. Para escribir contenido en el archivo:

```
arch.writeBytes( cadenaEntrada );
```

iv. Para cerrar el archivo:

```
arch.close();
```

Se recomienda revisar más detalles sobre archivos de acceso aleatorio y su uso en:
https://docs.oracle.com/javase/7/docs/api/java/io/RandomAccessFile.html

## 2. INTRODUCCIÓN AL USO DE BBDD

Cualquier aplicación computacional que manipule datos consta de al menos:

- La fuente de datos
- Aplicación que gestiona los datos

En general:

- La fuente de datos puede ser una BBDD, p. ej. Alguna construida en MySQL
- Aplicación, puede ser un programa Java que se conecte a la BBDD y procese los datos

### 2.1 Java y la conexión al BBDD

Para que una aplicación Java se conecte a una BBDD es necesario hacer que se comuniquen dos ambientes "que hablan" diferentes lenguajes.

- Uso de drivers y medios de conexión a las BBDD
- Las aplicaciones usando LDP con conexión a la BBDD

Los drivers son conjunto de clases que implementan las clases e interfaces del API JDBC (Java Data Base Conectivity) necesarias para que una aplicación Java pueda conectarse con una BBDD.

### 2.2 Consideraciones sobre JDBC

Se crea una interfaz, que permite a la aplicación Java comunicarse con las BBDD mediante un concepto similar al de componentes ODBC.

Las clases de objetos para iniciar la transacción con la BBDD:

- son 100% Java
- se permite mantener las características que el lenguaje posee: seguridad, robustez y portabilidad.

Esto hace que el JDBC esté diseñado para:

- ser independiente de la plataforma e
- incluso de la base de datos sobre la que se desee actuar.

### 2.3 Drivers JDBC

Existen 4 tipos de manejadores (o drivers) JDBC, que difieren en:

- si usan o no tecnología 100% Java,
- rendimiento y
- flexibilidad para cambiar de BBDD.

Estos 4 tipos de drivers son:

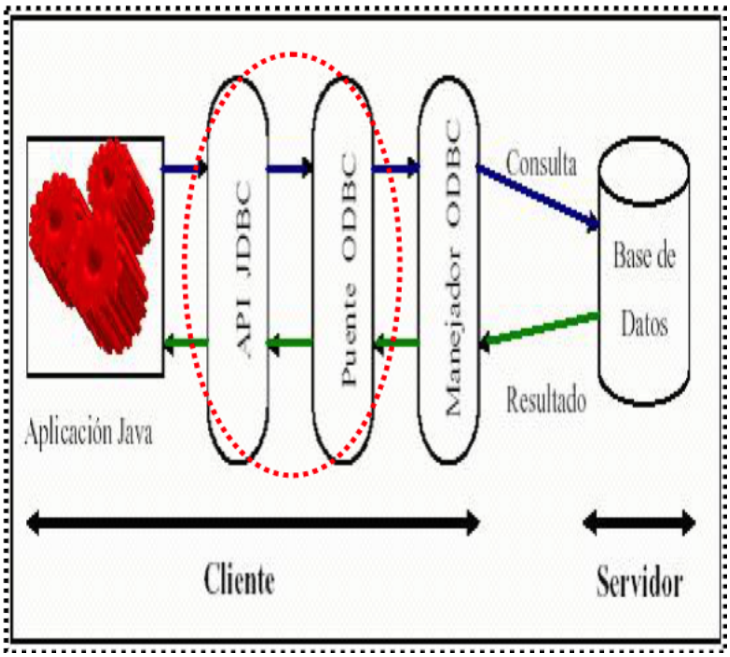
- Puente JDBC-ODBC (tipo1) (El que se utilizará para explicar el concepto en este apunte)
- API nativo (tipo2)
- JDBC-Net (tipo3)
- Protocolo Nativo (tipo4)

### 2.4 Driver JDBC-ODBC

Driver tipo puente JDBC-ODBC delega todo el trabajo sobre un manejador ODBC, que es quien realmente se comunica con la BD.

El puente JDBC-ODBC permite la conexión desde Java a BBDD que no proveen manejadores JDBC.

La siguiente figura muestra las capas que se ejecutan para que una aplicación Java se comunique con una BBDD usando el driver tipo JDBC-ODBC.




**2.5 Usando el driver JDBC-ODBC desde aplicaciones Java**

Para acceder a una BBDD mediante JDBC se necesitan:

- **url**, que es ruta donde está alojada físicamente la BBDD en el servidor
- Driver

Para poder trabajar con SQL, se necesita además hacer un import del package:

- import java.sql.*;

La **url** (Uniform Resource Locator) de JDBC identifica:

- una BBDD
- un protocolo de conexión a ésta.

Toda **url** de JDBC consta siempre de 3 partes:

**protocolo:subprotocolo:subnombre**

Analizando cada una de las partes que compone una **url** podemos decir que:
**protocolo:** es el protocolo de la conexión. En nuestro caso siempre es jdbc.

**subprotocolo**: identifica el tipo de mecanismo de conexión empleado para acceder a la BBDD.
**subnombre**: identifica la BBDD a la que nos deseamos conectar.

Para un ejemplo de URL de conexión, veamos el siguiente caso:

- **jdbc:odbc:bdTest**
  - **protocolo: jdbc**
  - **subprotocolo**: Identifica el tipo de mecanismo de conexión empleado para acceder a la BBDD. En este caso, el subprotocolo empleado será ODBC, por lo que el DriverManager intentará establecer la conexión con todos los manejadores registrados que sean de tipo 1, puente JDBC-ODBC.
  - **subnombre:** bdTest

El driver lo almacenamos en una variable tipo String:

- String **driver** = "sun.jdbc.odbc.JdbcOdbcDriver";

Además del driver, se requiere hacer uso de la interfaz Connection, la cual permite instanciar un objeto que representan consultas que se ejecutarán en la BBDD y permite acceder a la información que ésta contiene.

**Connection** conexión = null;
conexión = DriverManager.getConnection (url, <<userBBDD>>, <<pwdBBDD>> );

La otra interfaz requerida es **Statement**, la cual permite enviar y ejecutar instrucciones de tipo SQL en el motor de BBDD.

Se puede obtener un objeto que implemente esta interfaz a partir del método **Statement createStatement( )** de la interfaz Connection.

Para enviar una consulta de tipo SQL se emplea el método **executeQuery(String sql),** el cual devuelve un objeto tipo **Resulset**.

Por último, la interfaz **ResulSet** representa un conjunto de datos que son el resultado de una consulta SQL. Ésta posee una serie de métodos **getXXX(int columna)** y **getXXX(Stringcolumna),** que permiten acceder a los resultados de la consulta (para la sintaxis concreta de estos métodos acudir al javadoc de la interfaz ResulSet).

Para acceder a los distintos registros se usa un cursor, que inicialmente apunta justo antes de la primera fila. Para desplazar el cursor se usa el método **next( ).**

Ver el detalle de BBDD y SQL con Java y JDBC en sitio oficial de Oracle:
http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

Antes de terminar nuestra aplicación, se deben liberar los recursos de la conexión, una vez que se hayan terminado las operaciones deseadas sobre la BBDD. Esto con el fin de hacer más eficiente la gestión de los recursos y por temas de seguridad (acceso a datos, cierre de sesiones, etc.).

Los recursos que se deben liberar son el:

- Resultset,
- Statment y
- la propia conexión.

Para liberarlos se invoca al método **close( )** de sus correspondientes objetos.

Resumiendo un esquema de conexión a BBDD:
// se asume creada la BBDD
1. se referencia con el ODBC (para ocupar el driver puente JDBC-ODBC),
2. se crea la aplicación con los siguientes pasos principales:
    i. Establecer una conexión con la base de datos;
    ii. Ejecutar instrucciones SQL;
    iii. Si las intrucciones devuelven datos, procesarlos;
    iv. Liberar los recursos de la conexión;
    v. Luego se compila y listo, funciona de maravilla!!.

Una posible implementación del esquema anterior podría ser:

```
//..............
String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
String url = "jdbc:odbc:bdTest";
Statement statment = null;
Connection conexion = null;
ResultSet resulset = null;
//.....................
conexion = DriverManager.getConnection(url, "admin", "admin@X1");
statment = conexion.createStatement();
resulset = statment.executeQuery( "SELECT * from tblTest ");

while (resulset.next( ))  {
    System.out.print(resulset.getString("NOMBRE")+ ",");
}
//..............
resulset.close();
statment.close();
conexion.close();
```

¿Cómo se llama la BBDD?
¿Qué tabla usamos de la BBDD?
¿Qué datos se recuperan desde la BBDD?

**Resumiendo**

- Uso de archivos de acceso aleatorio con Java
- Uso de BBDD con Java
    - JDBC-ODBC
    - Características y consideraciones
    - Creando una conexión
    - Ejemplo