

PROTOCOLOS DE ACCESO A BBDD, JDBC

Javier García-Retamero Redondo

¿QUÉ PROTOCOLOS PODEMOS USAR?

- **ODBC (Conexión abierta a BD):** Define una API que pueden usar las aplicaciones para abrir una conexión con una BD. El servidor tiene que ser compatible ODBC.
- **JDBC (Conectividad de BD con Java):** Define una API que puedan usar los programas Java para conectarse a los servidores de BD.
- **OLE-DB (Enlace e incrustación de objetos para BD):** Es una API de Microsoft con objetivos parecidos a ODBC pero para orígenes de datos que no son BD (no necesariamente soportan SQL)
- **ADO (Objetos activos de BD):** Es una API de Microsoft con funcionalidad OLE-DB para ser llamada desde lenguajes como VBScript y Jscript.

PROTOCOLOS DE ACCESO A BBDD: JDBC

ENTENDIENDO CÓMO FUNCIONA



EN QUÉ CONSISTE

- Proporciona una **librería** para acceder a datos, principalmente provenientes de BD relacionales que usan SQL.
- Dispone de una interfaz distinta para cada BD, es lo que denominamos **driver** (controlador o conector).
- Consta de un conjunto de **clases e interfaces** que nos permiten escribir aplicaciones Java para gestionar las siguientes tareas con una BD relacional:
 - Conectarse a la BD
 - Enviar DML y DDL a la BD
 - Recuperar y procesar los datos recibidos

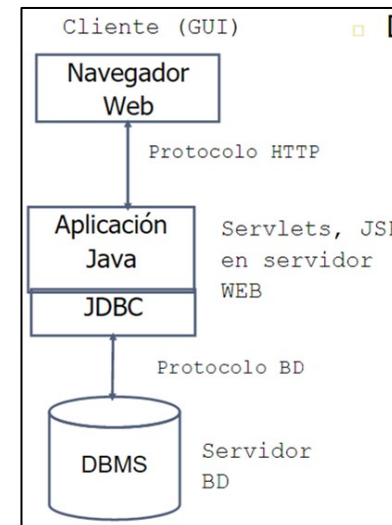
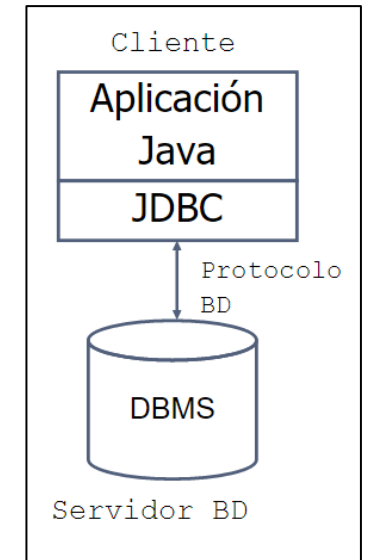
PROTOCOLOS DE ACCESO A BBDD: JDBC

ARQUITECTURA



DOS Y TRES CAPAS

- **Modelo de dos capas:**
La aplicación Java “habla” directamente con la BD a través de un **driver** JDBC que reside en el mismo lugar que la aplicación.
- **Modelo de tres capas:**
El **driver** JDBC reside en una máquina distinta.



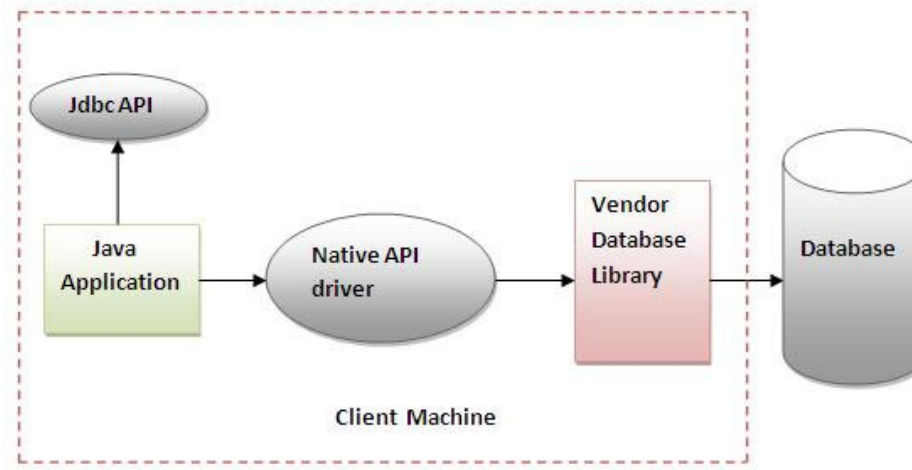
PROTOCOLOS DE ACCESO A BBDD: JDBC

TIPOS DE DRIVERS



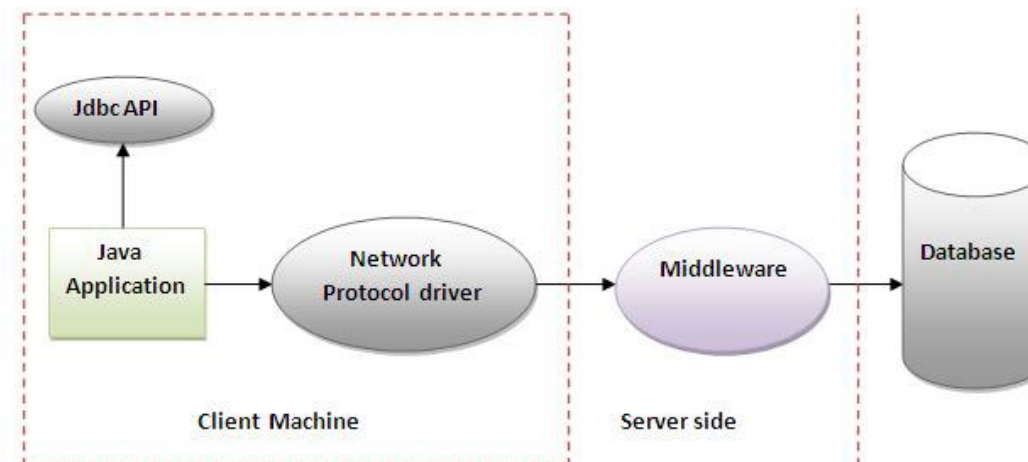
TIPOS DE DRIVERS

- **Native:**
 - Controlador escrito parcialmente en Java y código nativo
 - Traduce las llamadas de la API de JDBC en llamadas a la BD.
 - Se instala en el cliente código binario del cliente de BBDD y del SSOO



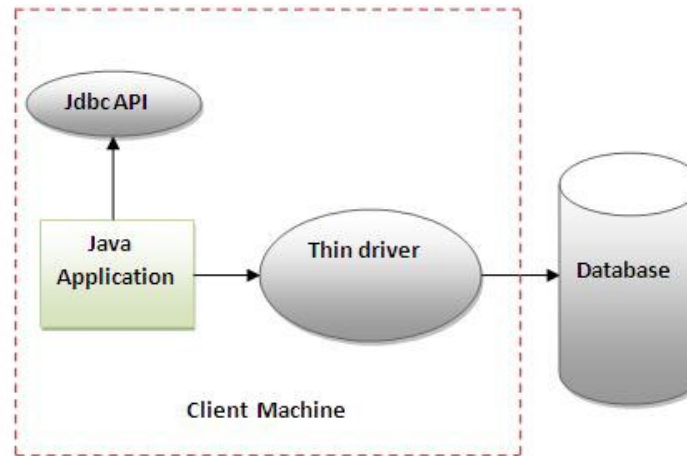
TIPOS DE DRIVERS

- **Network:**
 - Controlador Java puro
 - Utiliza un protocolo de red (por ejemplo http) para comunicarse con la BD.
 - Convierte:
 - llamadas al API de JDBC Java → llamadas propias del protocolo de red independiente de la BBDD → (Software intermedio, Middleware) al protocolo usado por el motor de BBDD
 - No en cliente



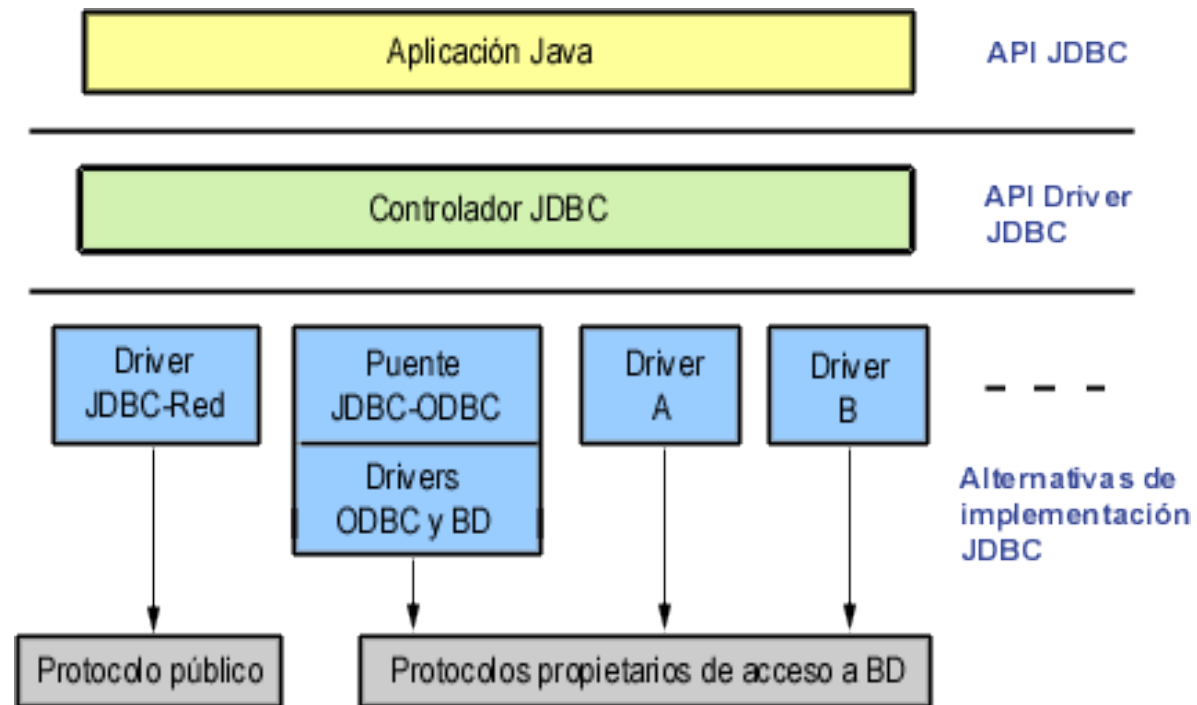
TIPOS DE DRIVERS

- **Thin:**
 - Controlador de Java puro con protocolo nativo
 - Traduce las llamadas de la API de JDBC → llamadas del protocolo de red de la BD.



Las formas más utilizadas son las dos últimas.

TIPOS DE DRIVERS



PROTOCOLOS DE ACCESO A BBDD: JDBC

CÓMO FUNCIONA



CÓMO FUNCIONA

- Utilizaremos el paquete `java.sql` con las siguientes clases e interfaces:

Clase	Función
Driver	Permite conectarse a un gestor de BD determinado
DriverManager	Permite gestionar todos los drivers instalados
DriverPropertyInfo	Proporciona información acerca de un driver
Connection	Representa una conexión con la BD. Puede haber varias
Statement	Permite ejecutar sentencias SQL sin parámetros
PreparedStatement	Permite ejecutar sentencias SQL con parámetros de entrada.
CallableStatement	Permite ejecutar sentencias SQL con parámetros de entrada/salida como llamadas a procedimientos almacenados
ResultSet	Contiene las filas resultantes de ejecutar una SELECT
ResultSetMetadata	Permite obtener información sobre un ResultSet, como el número de columnas, sus nombres, etc

CÓMO FUNCIONA

- Pasos para trabajar con la BBDD:

Pasos	Código
Añadir al proyecto el driver específico de la BBDD a la que nos queremos conectar	*.jar
Cargar el driver	<code>Class.forName("nombre_del_driver");</code>
Realizar la conexión a la BBDD	<code>Connection conexion = DriverManager.getConnection("cadena_conexión");</code>
Operaciones con los datos	
Cerramos la conexión	<code>conexion.close();</code>

CÓMO FUNCIONA

- Conexión a diferentes motores de BBDD (BBDD Embebidas)

Motor	Código
SQLite	<p>Añadir la librería:</p> <ul style="list-style-type: none">• Buscar librería en repositorio de Maven (https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc)• Encontrarás el código que debes añadir al archivo pom.xml del proyecto: <pre><dependencies> <dependency> <groupId>org.xerial</groupId> <artifactId>sqlite-jdbc</artifactId> <version>3.47.0.0</version> </dependency> </dependencies></pre>

CÓMO FUNCIONA

- Conexión a diferentes motores de BBDD (BBDD Embebidas)

Motor	Código
SQLite	<ul style="list-style-type: none">• Crearemos una clase que siga el patrón Singleton.• Definiremos los datos de conexión: <code>driver = "org.sqlite.JDBC";</code> <code>urlconnection = "jdbc:sqlite:./bbdd/ejemplo.db";</code>• Para conectar con la bbdd: <code>Class.forName(driver);</code> <code>this.conexion = DriverManager.getConnection(urlconnection);</code>• Tenemos que cerrar la conexión: <code>conexion.close();</code>



CÓMO FUNCIONA

- Conexión a diferentes motores de BBDD (BBDD no Embebidas)

Motor	Código
Oracle	<p>Añadir la librería:</p> <ul style="list-style-type: none">• Buscar librería en repositorio de Maven (Maven Repository: com.oracle.database.jdbc » ojdbc11)• Encontrarás el código que debes añadir al archivo pom.xml del proyecto: <pre><dependencies> <dependency> <groupId>com.oracle.database.jdbc</groupId> <artifactId>ojdbc11</artifactId> <version>23.5.0.24.07</version> </dependency> </dependencies></pre>

CÓMO FUNCIONA

- Conexión a diferentes motores de BBDD (BBDD no Embebidas)

Motor	Código
Oracle	<ul style="list-style-type: none">• Crearemos una clase que siga el patrón Singleton.• Definiremos los datos de conexión: <pre>driver = "oracle.jdbc.driver.OracleDriver"; urlconnection = "jdbc:oracle:thin:@localhost:1521/FREE";</pre>• Establecemos el usuario y la contraseña: <pre>this.propiedades = new Properties(); this.propiedades.setProperty("user", "dam2"); this.propiedades.setProperty("password", "dam2");</pre>• Para conectar con la bbdd: <pre>Class.forName(driver); this.conexion = DriverManager.getConnection(urlconnection, propiedades);</pre>• Tenemos que cerrar la conexión: <pre>conexion.close();</pre>