



Aplicaciones Telematicas

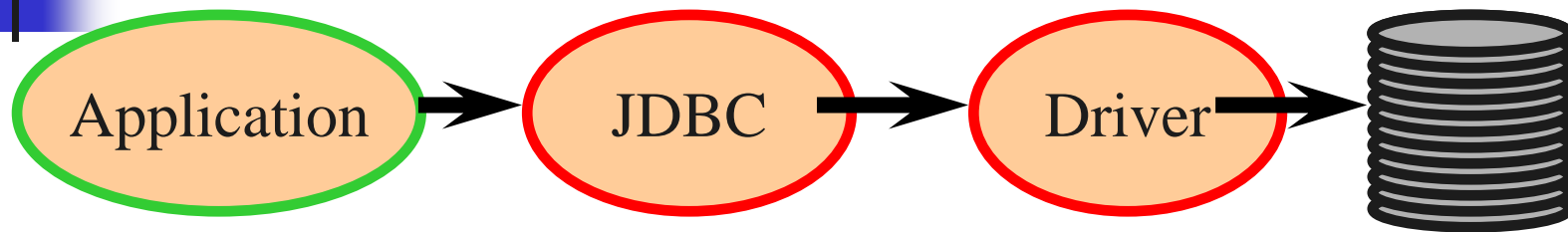
ACCESO A BASES DE DATOS
JDBC



Qué es JDBC?

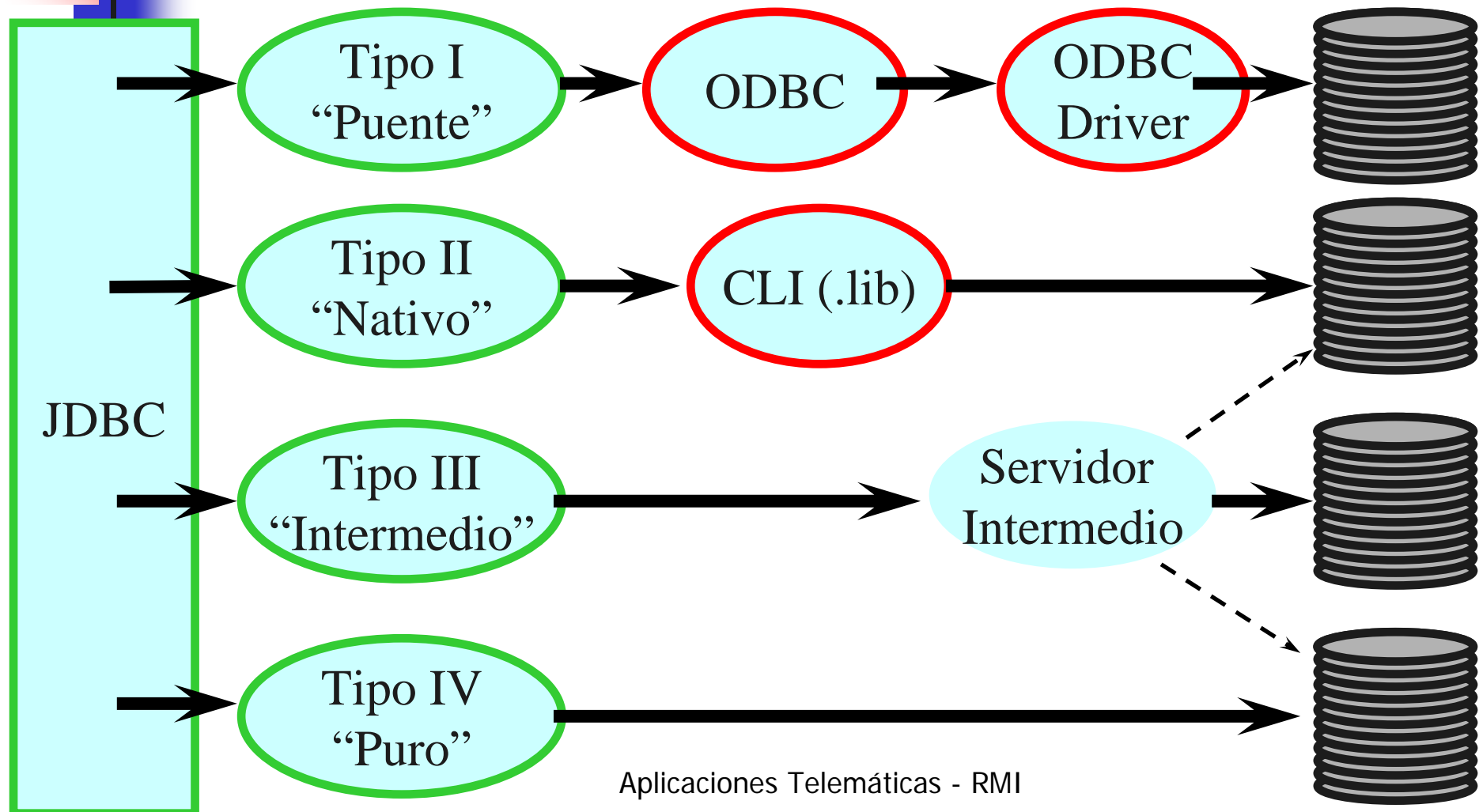
- ▶ JDBC es una especificación de un conjunto de clases y métodos de operación que permiten a cualquier programa Java acceder a sistemas de bases de datos.
- ▶ Driver → comunicación entre el API JDBC y la base de datos real.
- ▶ Conocimiento SQL – JAVA
- ▶ JDBC se implementa mediante clases del paquete:
java.sql

Arquitectura JDBC



- El código java llama a la librería JDBC
- JDBC carga un "driver"
- El driver dialoga con una particular base de datos
- Se pueden tener más de 1 driver -> más de 1 base de datos
- Objetivo: Poder cambiar el motor de base de datos sin cambiar el código de la aplicación

Tipos de Drivers JDBC





Driver Nativo

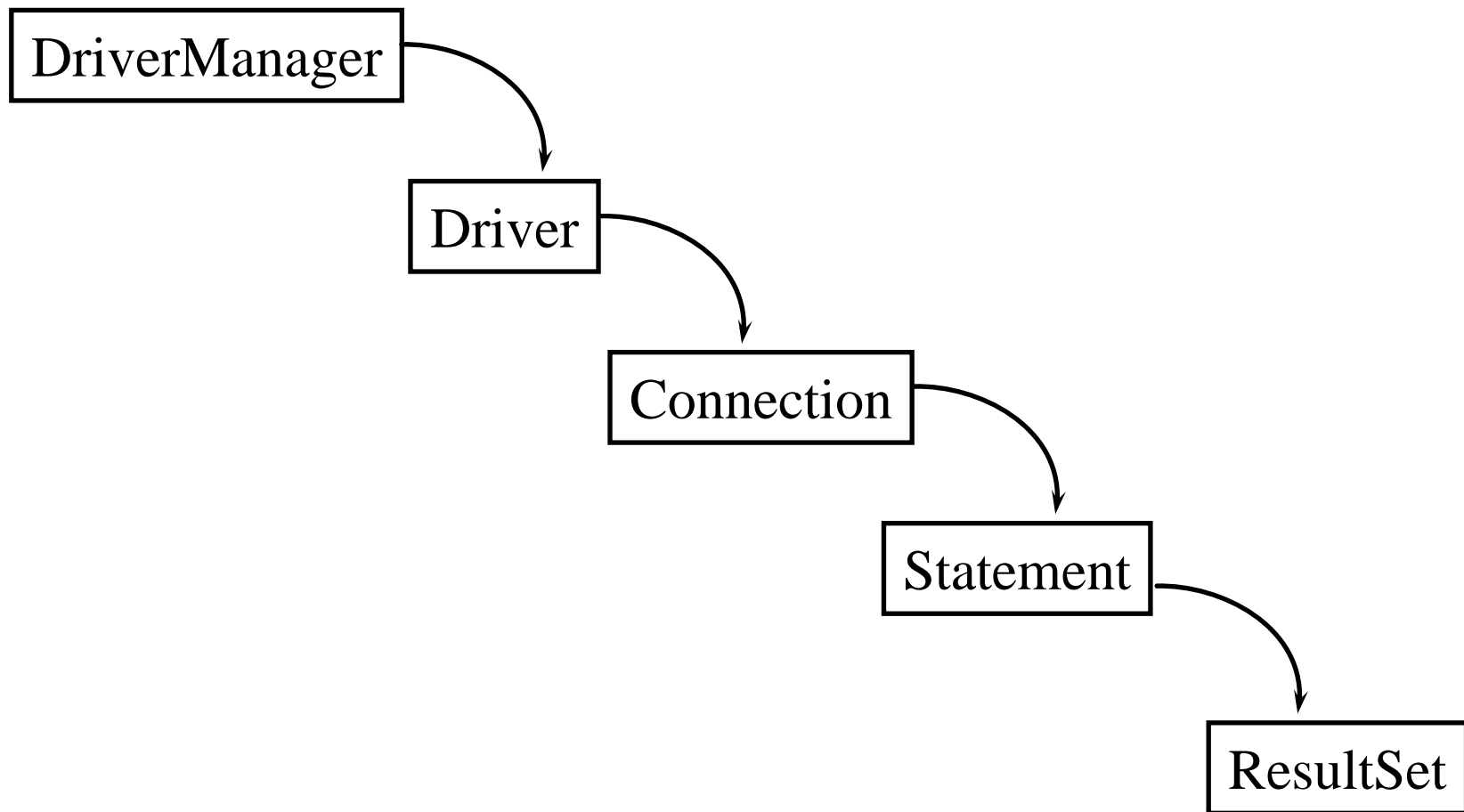
- 100 % Java
- Usa librerías de java para dialogar con la base de datos
- Desventaja: Se necesita cargar un driver por cada base de datos
- Ejemplos: Oracle, Mysql



JDBC (Clases de Objetos)

- DriverManager
 - Carga, elige drivers
- Driver
 - Se conecta con la base de datos actual
- Connection
 - Una serie de sentencias SQL hacia y desde la BD
- Statement
 - Un sentencia SQL
- ResultSet
 - Los registros obtenidos de la sentencia SQL

Uso de Clases JDBC





Statement Métodos

- `ResultSet executeQuery(String)`
 - Ejecuta una sentencia que retorna un sólo `ResultSet`
- `int executeUpdate(String)`
 - Ejecuta una sentencia `INSERT`, `UPDATE` o `DELETE`. Retorna el número de filas cambiadas
- `boolean execute(String)`
 - Ejecuta una sentencia SQL que puede retornar múltiples resultados (También se puede usar con sentencias del tipo: `DROP`, `INSERT`, `CREATE`, `DELETE` o `UPDATE`)



ResultSet Métodos

- `boolean next()`
 - Activa la siguiente fila
 - La primera llamada a `next()` activa la primera fila
 - Retorna falso si no hay más filas
- *Type* `getType(int columnIndex)`
 - Retorna el tipo del campo número `columnIndex` dentro de la tabla
 - Los campos se numeran a partir de 1 (no 0)
- *Type* `getType(String columnName)`
 - Igual que el anterior, pero usa el nombre del campo
 - Menos eficiente
- `int findColumn(String columnName)`
 - Busca el número de un columna



ResultSet Métodos (II)

- String getString(int columnIndex)
- boolean getBoolean(int columnIndex)
- byte getByte(int columnIndex)
- short getShort(int columnIndex)
- int getInt(int columnIndex)
- long getLong(int columnIndex)
- float getFloat(int columnIndex)
- double getDouble(int columnIndex)
- Date getDate(int columnIndex)
- Time getTime(int columnIndex)
- Timestamp getTimestamp(int columnIndex)



ResultSet Métodos (II)

- String getString(String columnName)
- boolean getBoolean(String columnName)
- byte getByte(String columnName)
- short getShort(String columnName)
- int getInt(String columnName)
- long getLong(String columnName)
- float getFloat(String columnName)
- double getDouble(String columnName)
- Date getDate(String columnName)
- Time getTime(String columnName)
- Timestamp getTimestamp(String columnName)



Mapeo de Tipos Java- SQL

<u>SQL type</u>	<u>Java Type</u>
■ CHAR, <u>VARCHAR</u> , LONGVARCHAR	String
■ <u>NUMERIC</u> , DECIMAL	java.math.BigDecimal
■ BIT	boolean
■ TINYINT	byte
■ SMALLINT	short
■ INTEGER	int
■ BIGINT	long
■ REAL	float
■ FLOAT, <u>DOUBLE</u>	double
■ BINARY, <u>VARBINARY</u> , LONGVARBINARY	byte[]
■ DATE	java.sql.Date
■ TIME	java.sql.Time
■ TIMESTAMP	java.sql.Timestamp



Código JDBC:

Establecer una conexión con la BD

- Cargar los drivers
 - `Class.forName("com.mysql.jdbc.Driver");`
- Hacer la conexión
 - `String url= ("jdbc:mysql://localhost/Agenda");`
 - `String usuario = "root";`
 - `String clave = "";`
 - `java.sql.Connection conexion = DriverManager.getConnection(url,usuario,clave);`
- Crear sentencias SQL
 - `Statement sentencia = conexion.createStatement();`



Código JDBC: Obtener campos de una tabla

- `String query = "SELECT * FROM AMIGOS" ;`
- `ResultSet rs = sentencia.executeQuery(query) ;`
- `while (rs.next()) {`
- `String nombre = rs.getString("NOMBRE") ;`
- `String apell = rs.getString("APELLIDOS") ;`
- `Date dat=rs.getDate("CUMPLE");`
- `System.out.println(nombre + " " + apell + " " + dat.toString()) ;`
- `}`



Lenguaje SQL

- Lenguaje estandarizado para consultar bases de datos (Structured Query Language)
- Independiente de la base de datos elegida (Supuestamente)



Lenguaje SQL (Inserciones)

- INSERT INTO tabla(campo1,campo2) VALUES (valor1, valor2)
 - Inserta un nuevo registro dentro de la tabla



Lenguaje SQL (Actualizaciones)

- UPDATE tabla SET (campo1= valor, campo2 = valor) WHERE condicion
- Actualiza la tabla asignando a los campos: campo1 y campo2 los valores valor1 y valor2 donde se cumpla la condicion



Lenguaje SQL (Borrado)

- DELETE FROM tabla WHERE condicion
- Elimina todos los registros dentro de tabla que cumplan la condicion



Lenguaje SQL (Selección)

- Obtener todos los campos de una tabla:
 - `SELECT * FROM tabla`
- Obtener todos los campos de una tabla (con condicion):
 - `SELECT * FROM tabla where condicion`
- Obtener campos de una tabla (con condicion):
 - `SELECT campo1,campo2 FROM tabla where condicion`
- Obtener campos de dos tablas (con condicion):
 - `SELECT tabla1.campo1,tabla1.campo2,tabla2.campo3 FROM tabla1,tabla2 where tabla1.campo1=tabla2.campo4`



Lenguaje SQL-Mysql (Notas)

- Los campos varchar siempre se referencian entre comillas simples:
 - `Select * from tabla where campo1='HOLA'`
- Los campos fecha tienen el formato: AAAA-MM-DD
 - `Select * from tabla where fecha>'2007-01-24'`
- Los campos numericos no necesitan comillas simples:
 - `Select * from tabla where edad>23`



SQL (Is Null)

- En SQL, NULL significa que el campo es vacío
- No es lo mismo que 0 o ""
- En JDBC, hay que preguntar expresamente si un campo es vacío invocando:
 - `ResultSet.isNull(column)`



Ejemplo

```
import java.sql.*;
public class PruebaMysql {
    static public void main( String[] args ) {
        Connection conexion;
        Statement sentencia;
        ResultSet resultado;
        System.out.println( "Iniciando programa." );
        // Se carga el driver JDBC-ODBC
        try {
            Class.forName( "com.mysql.jdbc.Driver" );
        } catch( Exception e ) {
            System.out.println( "No se pudo cargar el puente JDBC-ODBC." );
            return;
        }
        try {
            // Se establece la conexión con la base de datos
            conexion = DriverManager.getConnection("jdbc:mysql://156.35.130.139/APLI","alumno","alumno");
            sentencia = conexion.createStatement();
            try {
                // Se elimina la tabla en caso de que ya existiese
                sentencia.execute( "DROP TABLE AMIGOS" );
            } catch( SQLException e ) {};
```



Ejemplo (cont.)

```
// Esto es código SQL
sentencia.execute( "CREATE TABLE AMIGOS (" +
    " NOMBRE VARCHAR(15) NOT NULL, " +
    " APELLIDOS VARCHAR(30) NOT NULL, " +
    " CUMPLE DATETIME ) ");
sentencia.execute( "INSERT INTO AMIGOS " +
    "VALUES('JOSE','GONZALEZ','1974-03-23')");
sentencia.execute( "INSERT INTO AMIGOS " +
    "VALUES('PEDRO','GOMEZ','1961-06-09')");
sentencia.execute( "INSERT INTO AMIGOS " +
    "VALUES('GONZALO','PEREZ','1977-08-09')");

String query = "SELECT * FROM AMIGOS";
ResultSet rs = sentencia.executeQuery(query);
while ( rs.next() ) {

    String nombre = rs.getString("NOMBRE");
    String apell = rs.getString("APELLIDOS");
    Date dat=rs.getDate("CUMPLE");

    System.out.println( nombre + " " + apell + " " + dat.toString() );

}
} catch( Exception e ) {
    System.out.println( e );
    return;
}
}
System.out.println( "Creacion finalizada." );
}
}
```

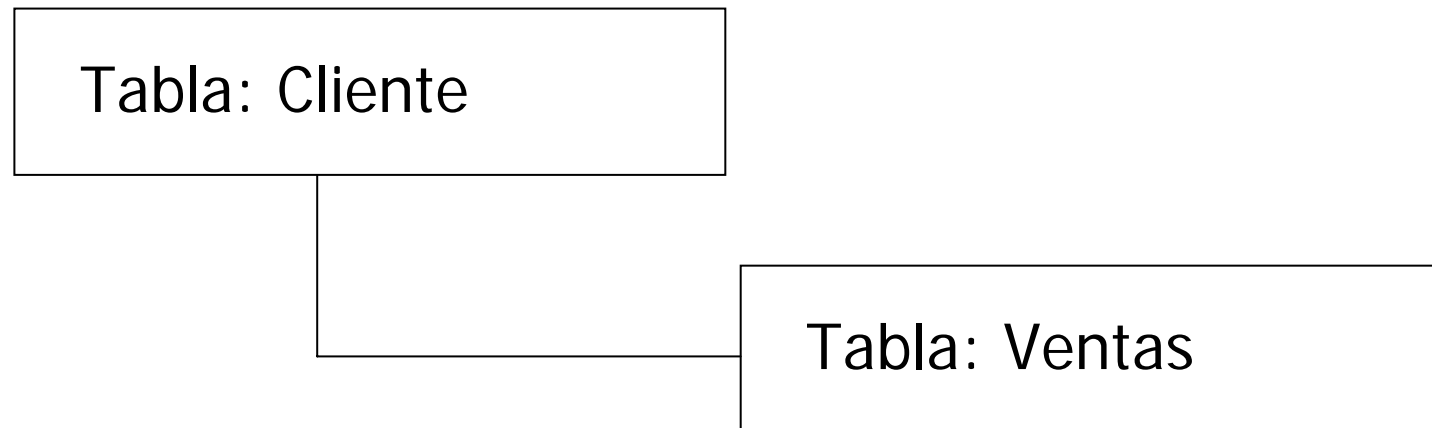


Compilación y Ejecución

- `Javac -classpath .\;. \mysql_driver.jar *.java`
- `Java -classpath .\;. \mysql_driver.jar Pruebamysql`

Ejercicios

1. Basándose en la estructura de base de datos:



Campos de Cliente:

CodigoCliente:	INT
Nombre:	Varchar(50)
Direccion:	Varchar(100)
Edad:	INT
Telefono:	Varchar(50)

Campos de Ventas:

CodigoCliente:	INT
Fecha:	DateTime
Producto:	Varchar(100)
Cantidad:	INT
Total:	Float



Ejercicios

- 1.- Obtener el nombre de todos los clientes que existan
- 2.- Obtener todos los datos de todos los clientes que existan
- 3.- Insertar un nuevo cliente y 3 ventas con datos inventados
- 4.- Obtener la suma total de ventas para ese cliente.
- 5.- Actualizar los datos del cliente y ponerle como edad 30 años.
- 6.- Borrar la última venta de ese cliente
- 7.- Mostrar un menú que permita:
 - 1.- Insertar un nuevo cliente
 - 2.- Insertar una venta para un cliente
 - 3.- Ver datos de un cliente (pidiendo código de cliente)
 - 4.- Ver datos de todas las ventas de un cliente (pidiendo código de cliente)