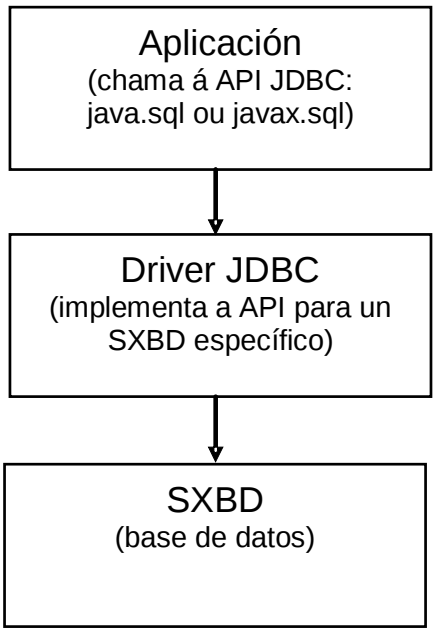
JDBC

Miguel Rodríguez Penabad
Laboratorio de Bases de Datos
Universidade da Coruña

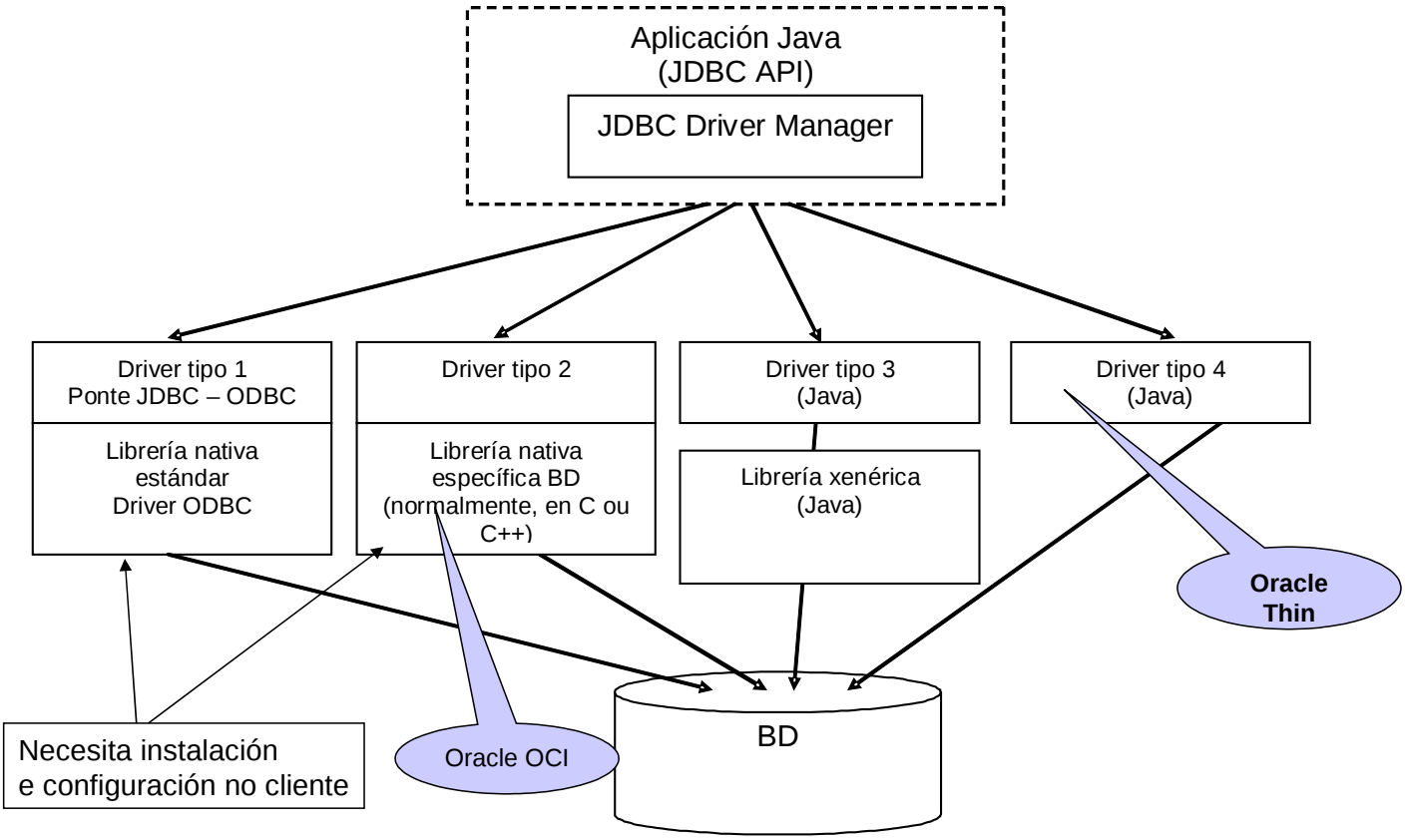
Índice

Introducción

- JDBC: Java DataBase Connectivity
- Similar a ODBC en MS Windows
- Permite a conexión de aplicacións Java a BDs relacionais
- Ofrece
  - API (Application Programming Interface) xenérica
  - Oculta a heteroxeneidade dos SXBD



Drivers JDBC

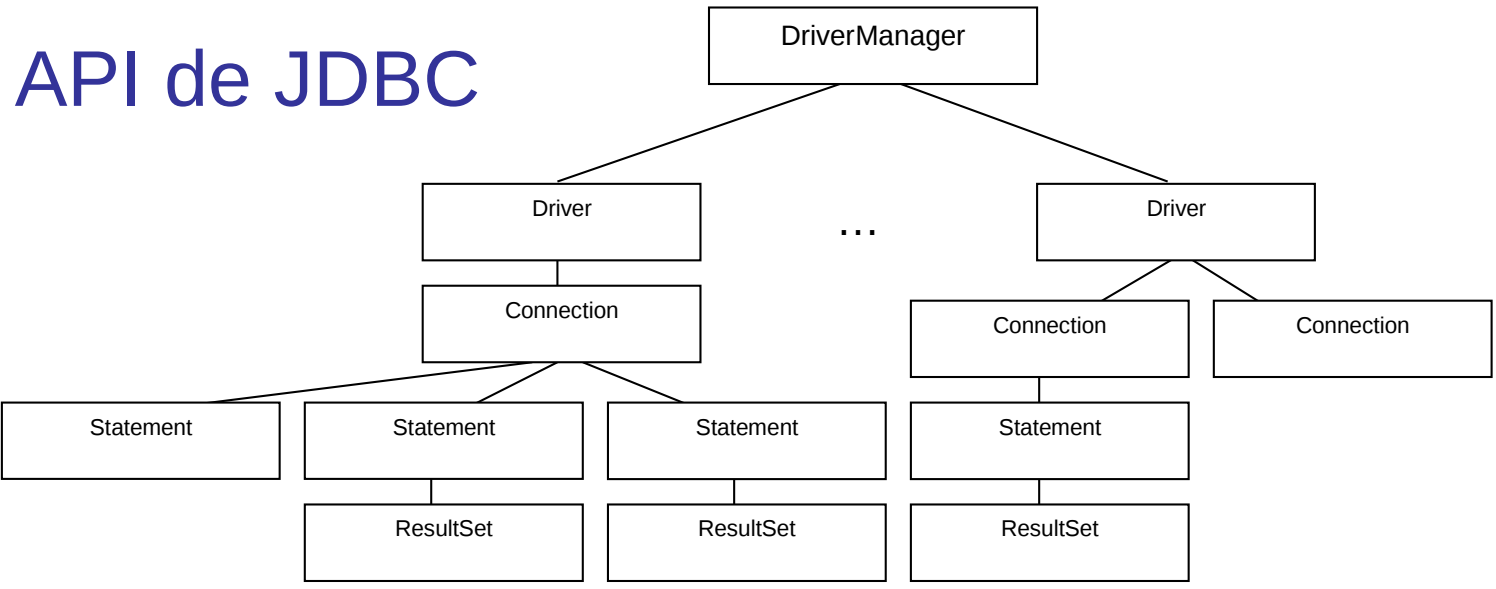


Drivers JDBC

- Tipos 1 e 2: Necesitan instalación ou configuración no cliente
- Tipo 3: Driver xenérico (tipo DBAnyWhere de DB2)
- Tipo 4: Driver 100% Java, particular para cada SXBD
  - Ficheiro JAR/Zip (no $CLASSPATH)
  - Distintos nomes

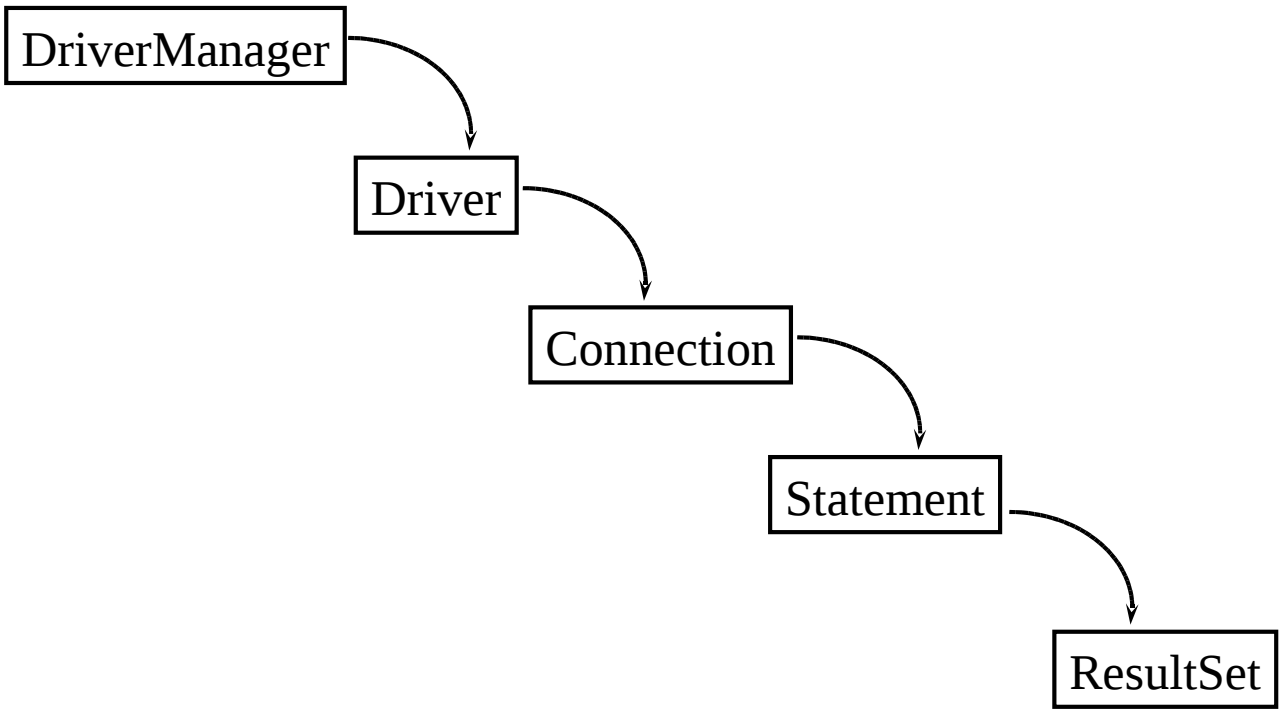
Para nós: `oracle.jdbc.OracleDriver` **en** `/usr/lib/oracle/11.1.0.1/client/lib/ojdbc5.jar`

API de JDBC



- Paquete `java.sql`
- Clases: Entre outras,
  - `DriverManager, Driver, Connection, Statement, PreparedStatement e ResultSet`

Clases na API de JDBC



Carga de drivers JDBC

```
Class.forName("oracle.jdbc.OracleDriver");
```

(Non necesitamos ter o driver JDBC presente en tempo de compilación)

Ou

```
DriverManager.registerDriver
        (new oracle.jdbc.OracleDriver ());
```

(Si que necesitamos o driver JDBC)

Conexión á BD

```
public static Connection
   getConnection(String url, String user, String
   password)                                    throws
   SQLException
```

Formato da URL:

jdbc:oracle:thin:@<EQUIPO>:<PORTO>:<SID>.

Exemplo

```
 String urlDriver =
"jdbc:oracle:thin:@10.10.7.46:1521:docenciadb";
 Connection conn = null;
 String user = “infabc00";
 String pass = “clave";
 String nomeDriver = "oracle.jdbc.OracleDriver";

 Class.forName(nomeDriver);
 conn = DriverManager.getConnection (urlDriver, user, pass);
```

Execución de sentencias

- Usamos métodos de Connection para crear obxectos de diversas clases
  - Statement (+ ResultSet)
  - PreparedStatement

Statement e Resultset

`Statement createStatement()`

- Statement: Usado para crear unha sentencia a partir dun string, que ó executarse devolverá un resultado

- Métodos de Statement:
  `ResultSet executeQuery(String` **`sentencia`**`)`
  - Executa a sentencia SQL e devolve un ResultSet.
  - ResultSet: Permite recorrer as filas resultantes (≈ cursor)

  `int executeUpdate(String` **`sentencia`**`)`
  - Para modificar datos ou para sentencias DDL. Devolve o número de filas modificadas (0 para DDL)

  `int execute(String` **`sentencia`**`)`
  - Executa sentencias arbitrarias (p.ex. De control)

Statement e Resultset (II)

- Métodos de ResultSet
  `boolean next()`
  - Obtén a primeira/seguinte fila, devolve false se non hai máis

  `void close()`
  - Libera (permite reusar) ResultSet

  `<Tipo> get<Tipo>(int indiceColumna)`
  `<Tipo> get<Tipo>(String nomeColumna)`
  - Obtén o valor (de tipo <Tipo>) da columna número indiceColumna ou de nome nomeColumna, na fila actual

  `Boolean isNull(int indiceColumna)`

Statement e Resultset (III)

```
import java.sql.*;
public class Exemplo
{
  public static void main(String[] args)
  {
    Connection conn = null;
   try {
        Class.forName("oracle.jdbc.OracleDriver");
        conn = DriverManager.getConnection
          ("jdbc:oracle:thin:@10.10.7.46:1521:docenciadb", “infabc00",
   “clave");

        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("Select ename, sal from emp");
        while (rs.next()){
           System.out.println(rs.getString(1) + "\t" + rs.getFloat(2));
        }
        rs.close(); stmt.close(); conn.close();
   } catch (Exception e) {
      e.printStackTrace();
   }
  }
}
```

PreparedStatement

`PreparedStatement prepareStatement(String` **`sentencia`**`)`

- Créase inicialmente coa sentencia SQL
- A sentencia *prepárase* (e optimízase) unha única vez
- Permite usar parámetros (con “?”)
  - `setInt(indice, <valor>)`
  - `setString(indice, <valor>)`
  - **`setDate(indice, <valor>)`**
  - …
- Execución: similar a Statement (pero sin argumentos)
  - `executeQuery()`
  - `executeUpdate()`
  - `execute()`

PreparedStatement

```
import java.sql.*;
public class Exemplo
{
  public static void main(String[] args)
  {
    Connection conn = null;
   try {
        Class.forName("oracle.jdbc.OracleDriver");
        conn = DriverManager.getConnection
          ("jdbc:oracle:thin:@10.10.7.46:1521:docenciadb", “infabc00", “clave");

        PreparedStatement ps =
                  conn.prepareStatement("Select ename, sal from emp where sal> ?");
        ps.setFloat(1,800.0)
        ResultSet rs = ps.executeQuery();
        while (rs.next()){
           System.out.println(rs.getString(1) + "\t" + rs.getFloat(2));
        }
        rs.close(); ps.close(); conn.close();
   } catch (Exception e) {
      e.printStackTrace();
   }
  }
}
```

Modificación de datos (I)

- Con Statement

```
statement.executeUpdate("create table t1("+
                        " a int, "+
                        " b char(5) "+
                        " c date)");
statement.executeUpdate(“insert into t1 " +
        " values(1, 'texto', '31/12/06') ");
```

Modificación de datos (II)

- Con PreparedStatement
  - Reusamos a sentencia
  - Non temos que formatear os datos dependendo do seu tipo

```
PreparedStatement ps =     conn.prepareStatement("insert into t1 " +
                     " values(?, ?, ?) ");
ps.setInt(1,1);
ps.setString(2,"Texto");
java.sql.Date data = /* Mellor usar Calendar – usa Time Zones */
            new java.sql.Date(new java.util.Date().getTime());
ps.setDate(3,data)
ps.executeUpdate();

ps.setInt(1,2);
ps.setString(2,"Text2");
ps.setDate(3,data)
ps.executeUpdate();
```

Excepcións

- Código Java:
  - try {…} catch { …}
    - Control de erros
  - try{…} finally {…}
    - Cando hai que facer algo **sempre**
    - `close()` dos obxectos implicados (statement, preparedstatement, resultset, connection) se non se necesitan máis.
- Excepción representativa
  - SQLException

Transaccións

- Métodos en Connection
  - `setAutoCommit(<true/false>)`
  - `commit()`
  - `setSavePoint(nome_savepoint)`
    `releaseSavePoint(nome_savepoint)`
  - `rollback()`,
    `rollback(nome_savepoint)`
  - `setTransactionIsolation(nivel_aillamento)`

```
TRANSACTION_NONE,
TRANSACTION_READ_UNCOMMITTED,
TRANSACTION_READ_COMMITTED,
TRANSACTION_REPEATABLE_READ,
TRANSACTION_SERIALIZABLE
```

Bibliografía/Documentación adicional

- Trail: JDBC(TM) Database Access
  http://java.sun.com/docs/books/tutorial/jdbc/

- Fernando Bellas Permuy. Tutorial de JDBC
  (Transparencias da asignatura Integración de Sistemas)
  http://www.tic.udc.es/is-java/Tema3-1.pdf