

Conexión Base de Datos

Lenguaje de Programación 2



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

Dr. Freddy Paz

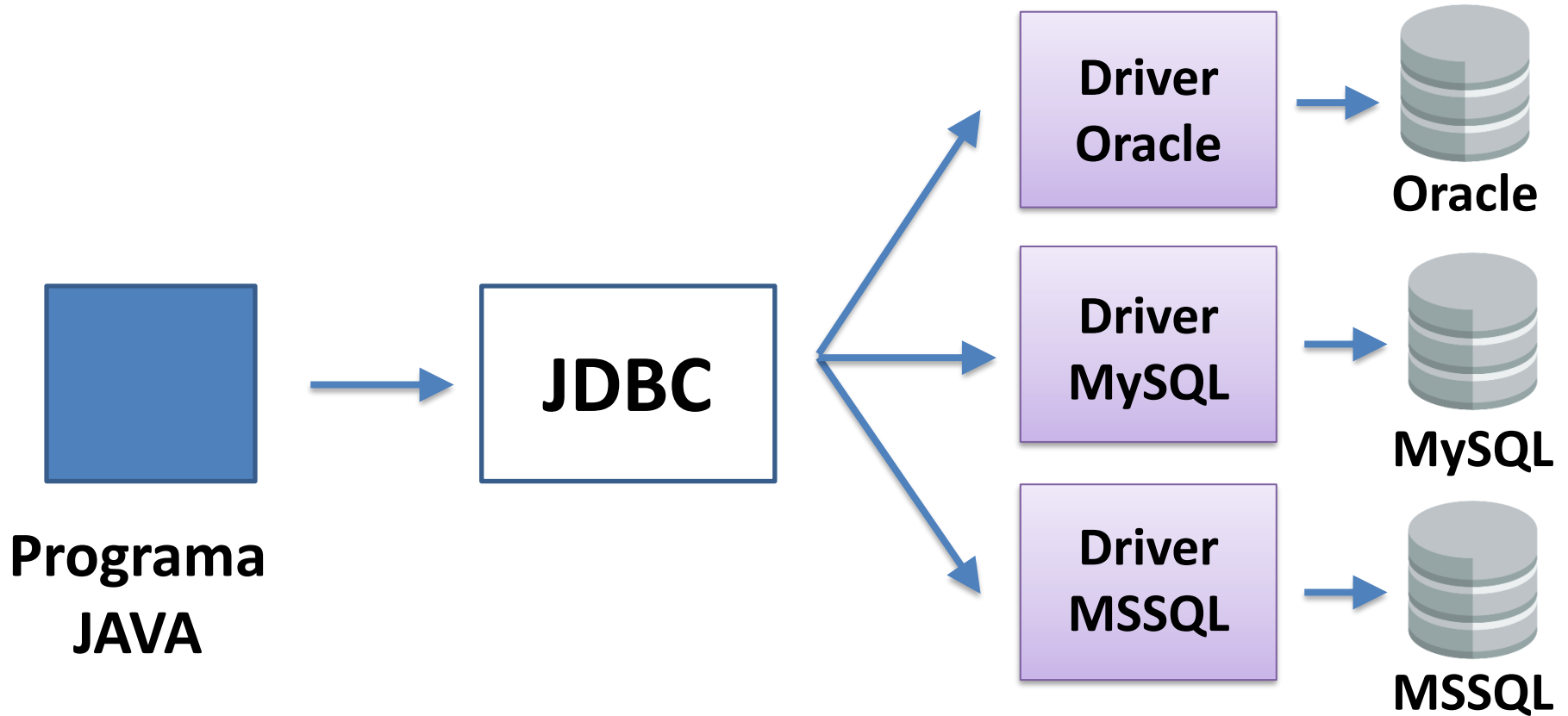
Conexión a Base de Datos

- Para establecer una conexión a base de datos es necesario utilizar una interface de aplicaciones (API) que permita acceder a los datos almacenados en **sistemas gestores de bases de datos** (DBMS) tanto relacionales como no relacionales, utilizando SQL (Lenguaje de Consulta Estructurado).

JAVA: JDBC

- ① **JDBC es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.**

Funcionamiento JDBC



JDBC Driver

- ① Un controlador o **driver JDBC** es un componente de software que permite conectar con bases de datos individuales. La interfaz JDBC requiere controladores (**drivers**) para cada base de datos. El controlador JDBC ofrece la conexión a la base de datos específica e implementa el protocolo para la transferencia de las consultas y resultados entre el cliente (aplicación) y la base de datos.

MySQL JDBC Driver

Nombre del Driver:

`com.mysql.jdbc.Driver`

Formato URL:

`jdbc:mysql://hostname/databaseName`



Connection

Un objeto *Connection* representa una conexión a una base de datos. La forma estándar de establecer una conexión con una base de datos es llamando al método **DriverManager.getConnection**. Este método toma como parámetro una cadena de caracteres que contiene una URL, usuario y password.

```
Connection con =  
DriverManager.getConnection(url, usuario,  
password)
```



Estableciendo la conexión

```
try {  
    //registrar el Driver  
    Class.forName("com.mysql.jdbc.Driver");  
  
    //establecer la conexión  
    Connection con =  
    DriverManager.getConnection("jdbc:mysql://50.62.209.73/in  
f282", "inf282", "inf282lp2");  
  
} catch (Exception ex) {  
    System.out.println(ex.getMessage());  
}
```



Estableciendo la conexión

Driver Versión: 8.0.11 (mysql-connector-java-8.0.11)

```
try {  
    //registrar el Driver  
    Class.forName("com.mysql.cj.jdbc.Driver");  
  
    //establecer la conexión  
    Connection con =  
    DriverManager.getConnection("jdbc:mysql://50.62.209.73/in  
f282?useSSL=false", "inf282", "inf282lp2");  
  
} catch (Exception ex) {  
    System.out.println(ex.getMessage());  
}
```



Statement

Un objeto *Statement* se usa para enviar sentencias SQL a una base de datos. Una vez que se ha establecido una conexión con una base de datos particular, esa conexión puede ser usada para enviar sentencias SQL. Un objeto *Statement* se crea con el método *createStatement* de *Connection* como en el siguiente fragmento de código:

```
Statement stmt = con.createStatement();
```



Statement

El método **executeQuery(sqlString)** es para sentencias SQL tipo SELECT.

El método **executeUpdate(sqlString)** es para sentencias SQL tipo INSERT, UPDATE, DELETE.



Statement - Insert

```
Statement sentencia = con.createStatement();  
String query = "INSERT INTO empleado  
(dni,nombres,apellido_paterno,apellido_materno) "  
              + "values "  
              + "('12114689','Jorge','Mendoza','Lopez')";  
int i = sentencia.executeUpdate(query);  
con.close();
```



Statement - Update

```
Statement sentencia = con.createStatement();
```

```
String query =
```

```
    "UPDATE empleado SET nombres = 'Karla Celeste' "
```

```
    + "WHERE "
```

```
    + "dni = '18276221'";
```

```
int i = sentencia.executeUpdate(query);
```

```
con.close();
```



Statement - Delete

```
Statement sentencia = con.createStatement();
```

```
String query =
```

```
        "DELETE FROM empleado WHERE dni =  
'18276221'";
```

```
int i = sentencia.executeUpdate(query);
```

```
con.close();
```



ResultSet

Un **ResultSet** contiene todos los registros (filas) que satisfacen las condiciones impuestas en una sentencia SQL y proporciona acceso a los datos en dichos registros a través de un conjunto de métodos **get** que permiten acceder a los diferentes campos o atributos (columnas) del registro actual. Un **ResultSet** mantiene un cursor que apunta al registro actual. El método **ResultSet.next()** se usa para moverse al siguiente registro del **ResultSet**, haciendo el siguiente registro el registro actual.



Statement - SELECT

```
Statement sentencia = con.createStatement();
String query =
    "SELECT * FROM empleado";
ResultSet rs = sentencia.executeQuery(query);
while(rs.next()){
    String dni = rs.getString("dni");
    String nombres = rs.getString("nombres");
    String apellido_paterno = rs.getString("apellido_paterno");
    String apellido_materno = rs.getString("apellido_materno");
    System.out.println(dni + " " + nombres + " " + apellido_paterno + " " +
        apellido_materno);
}
con.close();
```



Estableciendo la conexión

```
try {  
    //registrar el Driver  
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
    //establecer la conexión  
    Connection con =  
    DriverManager.getConnection("jdbc:sqlserver://184.168.194.78;  
    databaseName=LP2;integratedSecurity=false;username=fpaz;  
    password=123456;");  
} catch (Exception ex) {  
    System.out.println(ex.getMessage());  
}
```



Referencias



- D.J. Barnes y M. Kölling, Programación orientada a objetos con Java. Pearson Educación, 2007
- T. Budd, An introduction to Object-Oriented Programming (Third Edition). Pearson Education, 2001
- E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994
- B. Stroustrup, The C++ Programming Language (Third Edition) Addison-Wesley, 1997
- Agustín Froufe. Java 2. Manual de usuario y tutorial. Ed. Ra-Ma
- J. Sánchez, G. Huecas, B. Fernández y P. Moreno, Iniciación y referencia: Java 2. Osborne McGraw-Hill, 2001.
- B. Meyer, Object-Oriented Software Construction (Second Edition). Prentice Hall, 1997.