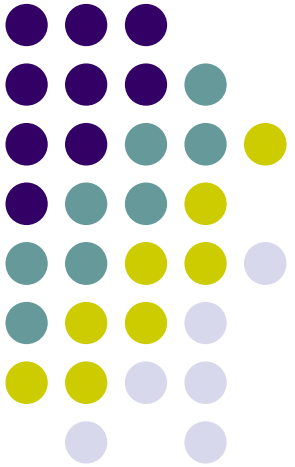
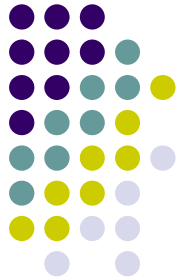
# Servlet – JSP - JDBC

Conectando Java al mundo de las Base de Datos



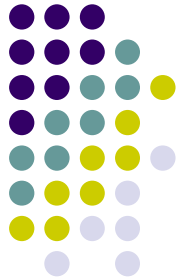
Lic. Claudio Zamoszczyk
claudio@honou.com.ar

# Contenidos



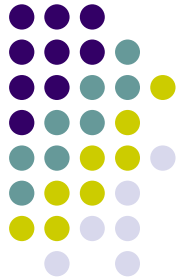
- ¿Qué es JDBC?
- ¿Por qué usar JDBC?
- Estructura del API JDBC
- ¿Cómo usar JDBC?
- ¿Cómo usar JDBC con JSP?
- ¿Cómo usar JDBC con Servlets?
- Pool de conexiones

# ¿Qué es JDBC?



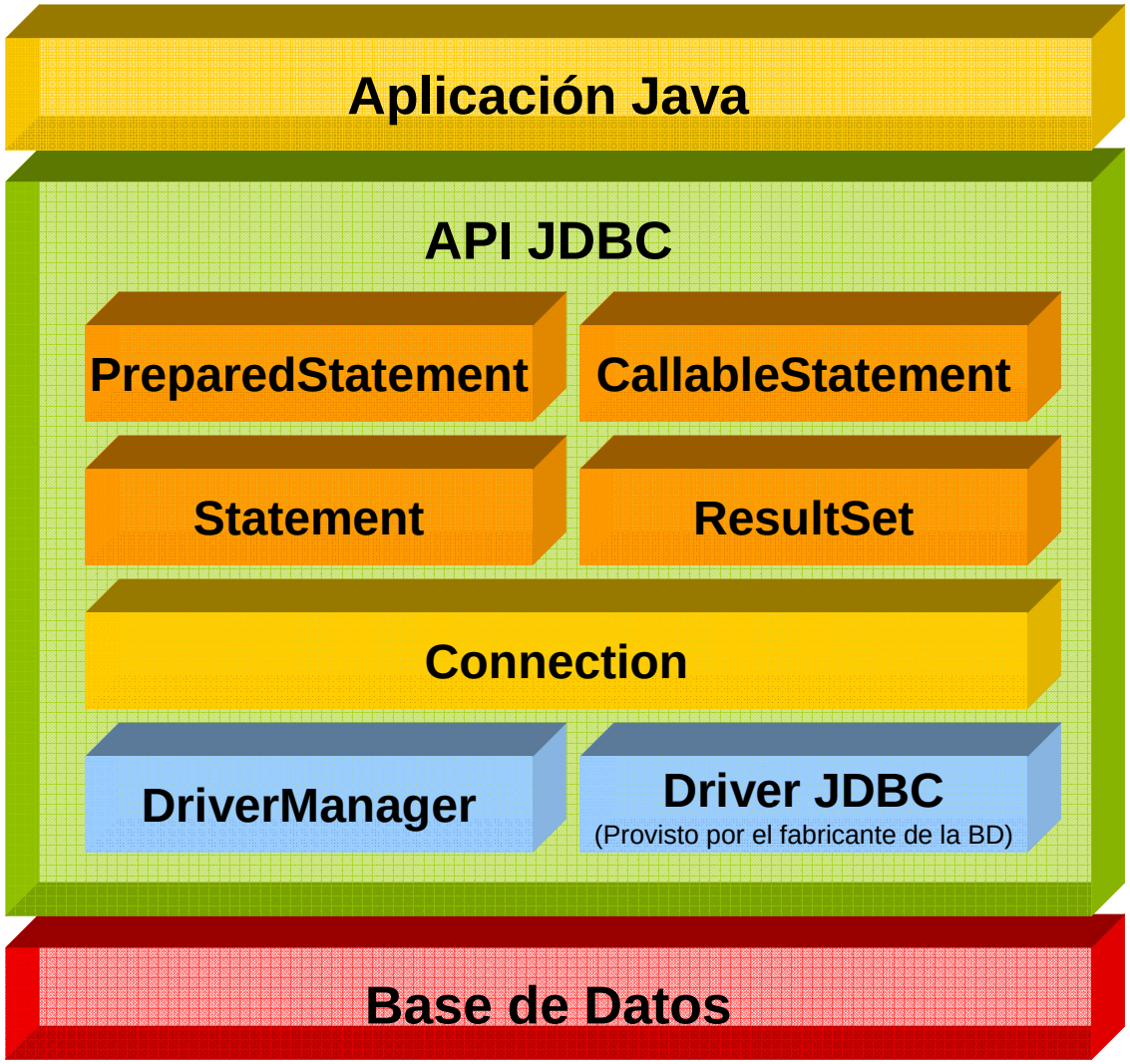
- **JDBC es una API, formada por conjunto de clases e interfaces, que permiten ejecutar sentencias SQL sobre una base de datos (externa) utilizando Java.**
- **JDBC = Java Database Connectivity**

# ¿Por qué usar JDBC?



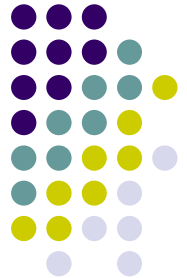
- **Ofrece un estándar de conexión a cualquier base de datos disponible en el mercado (Oracle, MySql, DB2, Sql Server, Informix, etc.).**
- **Permite establecer una conexión a una base de datos de manera simple.**
- **Permite enviar y ejecutar sentencias SQL (Select, Update, Insert, etc).**
- **Permite procesar los resultados de estas sentencias.**
- **Permite ejecutar Store Procedures**

# Estructura básica del API JDBC



# ¿Cómo usar JDBC?

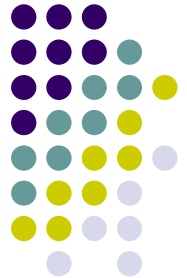
## Drivers



- Los Driver JDBC son los componentes que permite conectar a las diferentes BD con Java.
- Cada fabricante es el encargado de proveer el driver correspondiente, y este debe ser compatible con la especificación JDBC.
- Existen 4 tipos de Drivers JDBC.

# ¿Cómo usar JDBC?

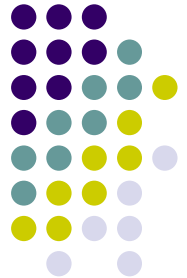
## Drivers



- Los Driver JDBC son los componentes que permite conectar a las diferentes BD con Java.
- Cada fabricante es el encargado de proveer el driver correspondiente, y este debe ser compatible con la especificación JDBC.
- Existen 4 tipos de Drivers JDBC.

# ¿Cómo usar JDBC?

## Cargando Drivers (Clas.for(…..))



- **Es necesario primero cargar una clase con el driver de la base de datos (esto lo provee el fabricante de la DB)**
- **Ejemplo:**

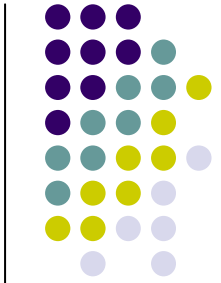
  **Class.forName("com.informix.jdbc.IfxDriver");**

  **Calss.forName("com.novell.sql.LDAPDriver");**

  **Class.forName("com.mysql.jdbc.Driver");**
- **Esto es particular según la base de datos que se usa**

# ¿Cómo usar JDBC?

## Estableciendo la Conexión (Connection)



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

- **Url de conexión**

**jdbc:mysql://localhost/test**

**jdbc:oracle://oraserver/db**

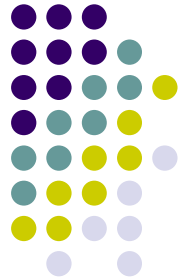
**jdbc:odbc:mydatabase**

**jdbc:vendor//servidor:puerto/bd…**

**Ver ejemplo primeraConexion.java**

# ¿Cómo usar JDBC?

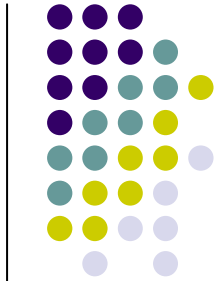
## Ejecutando sentencias SQL



- **JDBC permite enviar cualquier tipo de sentencia SQL. Aunque ésta fuera dependiente de la base de datos sólo se correría el riesgo de incompatibilidad al cambiar de base de datos.**

# ¿Cómo usar JDBC?

## Ejecutando sentencias SQL (Statement)



- **Statement stmt = conexion.createStatement();**
- **Métodos de Statement:**

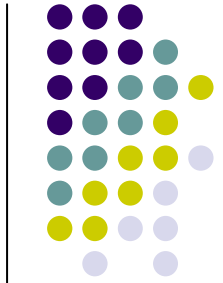
  **- stmt.execute(String), usado para ejecutar cualquier tipo de comando. Retorna true o false.**

  **- stmt.executeUpdate(String), usado para crear/modificar tablas, típicamente para create, update, delete. Retorna el numero de fila afectadas.**

  **- stmt.executeQuery(String) pusado para hacer consultas, retorna un cojunto de filas representadas en un objeto de la clase ResultSet, típicamente para select.**

- **Ver ejemplo PrimerComandoSql.java**

# ¿Cómo usar JDBC?
## Ejecutando consultas SQL (ResultSet)



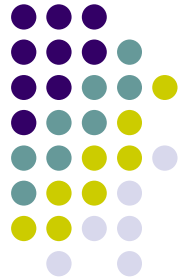
- **ResultSet rs = stmt.executeQuery(“select * from Personas”);**

```
while (rs.next()) {
    String s = rs.getString(“nombre”);
    int y = rs.getInt(“dni”);
    System.out.println(s + ”   “+ y );
}
```

- **Ver ejemplo PrimeraConsulta.java**

# ¿Cómo usar JDBC?

## Ejecutando sentencias precompiladas SQL (PrepareStatement)



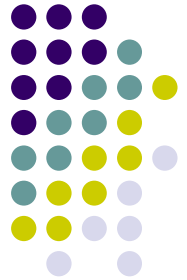
- Donde se ha usado Satement es generalmente posible usar PreparedStatement para hacer más eficientes las consultas.
- Una instrucción con PreparedStatement va a ser, en la mayoría de los casos, traducida a una consulta SQL nativa de la base de datos en tiempo de compilación.
- La otra ventaja es que es posible usar parámetros dentro de ella, pudiendo hacer más flexibles las consultas.

- PreparedStatement us = con.prepareSatatement("update alumnos set nombre = ?  where direccion like = ?");
  us.setString(1,"Juan Carlos")
  us.setString(2,"Callao 9876541236");

- Ver ejemplo PrimerPrepareStatement.java

# ¿Cómo usar JDBC?

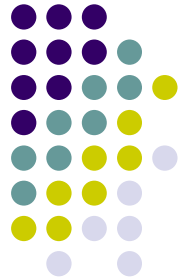
## Manejo de transacciones



- **Una transacción consiste en una o más sentencias que han sido ejecutadas y luego confirmadas (commit) o deshechas (rolled back)**
- **Auto-commit está preseteado.**
- **Si Auto-commit está desactivado se debe usar los métodos “commit” o “rollback” explícitamente.**

# ¿Cómo usar JDBC?

## Manejo de transacciones

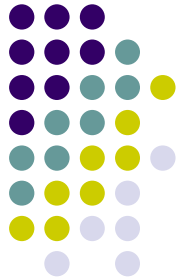


- **Para hacer uso de transacciones debe primero etearse el auto-commit a false**

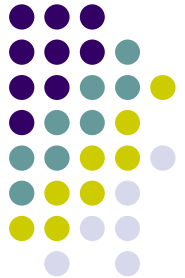
```
con.setAutoCommit(false)
PreparedStatement ps = .....
....
ps.executeUpdate()  ....
ps.excecuteUpdate() ...
con.commit();
```

- **Ver ejemplo ManejoDeTransacciones.java**

# ¿Preguntas?



# ¿Cómo usar JDBC con JSP?



**1 – Copiar el archivo .jar correspondientes al driver JDBC en el directorio /commons/lib o en el directorio de la app web /WEB-INF/lib**

**2 – Importar el paquete java.sql.***

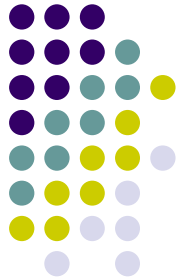
```
<%@ page language="java" import="java.sql.*"
contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-
8859-1"%>
```

**3 – Registrar el driver y crear una nueva conexión**

```
<%
Class.forName("com.mysql.jdbc.Driver");
Connection conexion =
DriverManager.getConnection("jdbc:mysql://localhost/productosdb","r
oot",“xxxx");
 %>
```

**4 – Luego de utilizar la conexión cerrar la misma**

# ¿Cómo usar JDBC con Servlets?



**1 – Copiar el archivo .jar correspondientes al driver JDBC en el directorio /commons/lib o en el directorio de la app web /WEB-INF/lib**

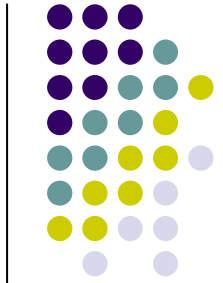
**2 – Importar el paquete java.sql.***

**3 – Registrar el driver y crear una nueva conexión**

```
Class.forName("com.mysql.jdbc.Driver");
Connection conexion =
DriverManager.getConnection("jdbc:mysql://localhost/productosdb","r
oot",“xxxx");
```

**4 – Luego de utilizar la conexión cerrar la misma**

# Concepto de pool de conexiones

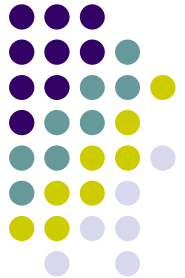


**Una orientación básica a JDBC implica que se realiza una conexión a la base de datos en cada servlet/jsp. Se repite el esquema conexión-operación-desconexión. Esta forma de trabajar es perfectamente válida, pero resulta ineficiente, ya que se están desperdiciando ciclos de CPU y memoria en cada conexión y desconexión.**

**En vez de esta orientación hay otra alternativa: crear un pool de conexiones, es decir, mantener un conjunto de conexiones ya abiertas y reutilizarlas según corresponda.**

**Cuando un servlet/jsp requiere una conexión la solicita al conjunto de conexiones disponibles y cuando no la va a usar la devuelve al pool. De esta forma nos ahorramos el consumo de tiempo de conexión y desconexión. La mayor parte de servidores de aplicaciones (Tomcat, WebSphere, etc.) incluyen clases que implementan un pool de conexiones.**

# Pasos para configurar el pool de conexiones



**1 – Copiar el archivo .jar correspondientes al driver JDBC en el directorio /commons/lib**

**2 – Definir en el archivo server.xml la configuración del contexto a utilizar con el pool y los parametros del pool**

**3 – Definir en el archivo web.xml la referencia de que pool de conexión se utilizara en la app. web**