



# Java con Bases de Datos

# Java con bases de datos

## ■ Requisitos previos

- ◆ Conocimientos de programación avanzada en Java
- ◆ Modelado de datos
- ◆ SQL

## ■ Objetivos

- ◆ Introducción al acceso a bases de datos desde aplicaciones con Java Data Base Connectivity (JDBC)

## ■ Dirigido a


- ◆ Programadores y Analistas/Programadores que deseen desarrollar aplicaciones para Internet con acceso a bases de datos

# Contenidos

- **Introducción**
- **El estándar JDBC**
- **Ejemplo con JDBC**
- **Clases de JDBC**
- **API JDBC 2.0**

# Introducción

## Tecnología de acceso a bases de datos

- ⊕ **Entusiasmo inicial de la tecnología Java como plataforma cliente/servidor y computación en Web**
  - ⊕ **Desarrolladores de bases de datos (Oracle, Informix) pretenden que los *applet* accedan a sus bases de datos a través de redes con tecnología Internet**
- 
- ⊕ **En la conferencia JavaOne (mayo 1996) Sun anuncia nuevos *APIs Enterprise* dirigidos a aplicaciones cliente/servidor**
    - ⊕ **JDBC (*Java DataBase Connectivity*) para acceso a bases de datos**
    - ⊕ **JVM incorpora un gestor específico para cada tipo de SGBD**
      - ⊕ **Se apoya en la especificación y filosofía de ODBC: JDBC ⇒ ODBC**

# Introducción

## Acceso a bases de datos desde Java

- ⊕ **API Java Enterprise**

  - ⊕ **JDBC**

- ⊕ **Métodos específicos proporcionados por los desarrolladores de cada base de datos**

- ⊕ **Métodos desarrollados por terceras compañías**

# El estándar JDBC

## ⊕ JDBC como API Enterprise

- ⊕ Conjunto de clases de acceso a bases de datos relacionales
- ⊕ Desarrollo de aplicaciones cliente/servidor dirigidas a empresa mediante objetos Java, applets y servlets
  - ⊕ Sistemas de facturación
  - ⊕ Reserva de billetes de avión
  - ⊕ Catálogos
  - ⊕ Marketing
- ⊕ Sistemas basados en una arquitectura de tres niveles
  - ⊕ Base de datos
  - ⊕ Lógica de la aplicación
  - ⊕ Interfaz de usuario

# El estándar JDBC

## Características

- ⊕ **Es parte de Java 1.1**
- ⊕ **Independiente de la plataforma**
- ⊕ **Independiente de la base de datos**
- ⊕ **Modelado en base a ODBC**
  - ⊕ **Disminuye la curva de aprendizaje por su amplia utilización**
  - ⊕ **Existen implementaciones eficaces de ODBC en casi todas las plataformas y para casi todas las bases de datos**
- ⊕ **Se basa en abstracciones comunes**
  - ⊕ **La conexión: *conexion***
  - ⊕ **El conjunto de resultados: *ResultSet***

# El estándar JDBC

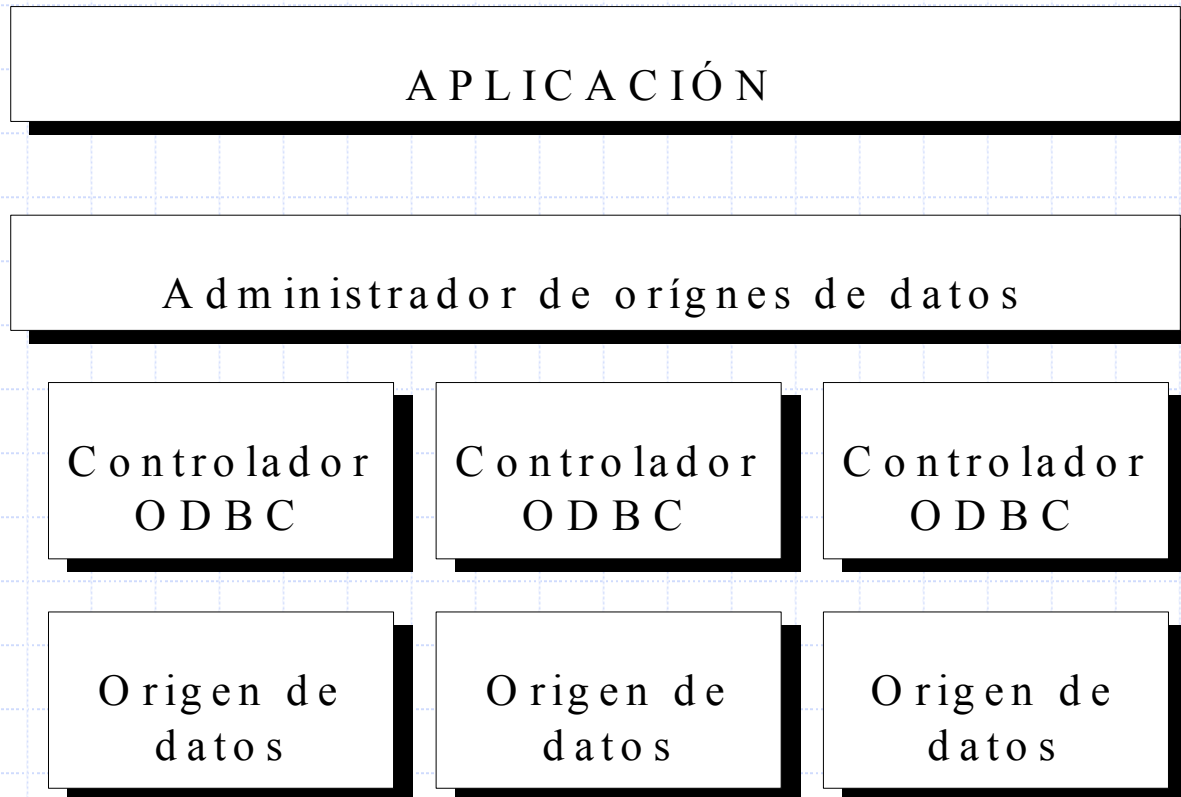
## Ventajas derivadas de JDBC:ODBC

- ⊕ **Facilita la comunicación de la aplicación con distintos SGBD's.**
  
- ⊕ **Proporciona una serie de funciones para la manipulación de datos (inserción, borrado y modificación), consultas, vistas y llamadas a procedimientos.**
  
- ⊕ **Presenta una arquitectura de cuatro niveles:**
  - ⊕ **Aplicación**
  - ⊕ **Administrador de orígenes de datos.**
  - ⊕ **Controlador/es ODBC**
  - ⊕ **Orígenes de datos**

# El estándar JDBC

## ODBC

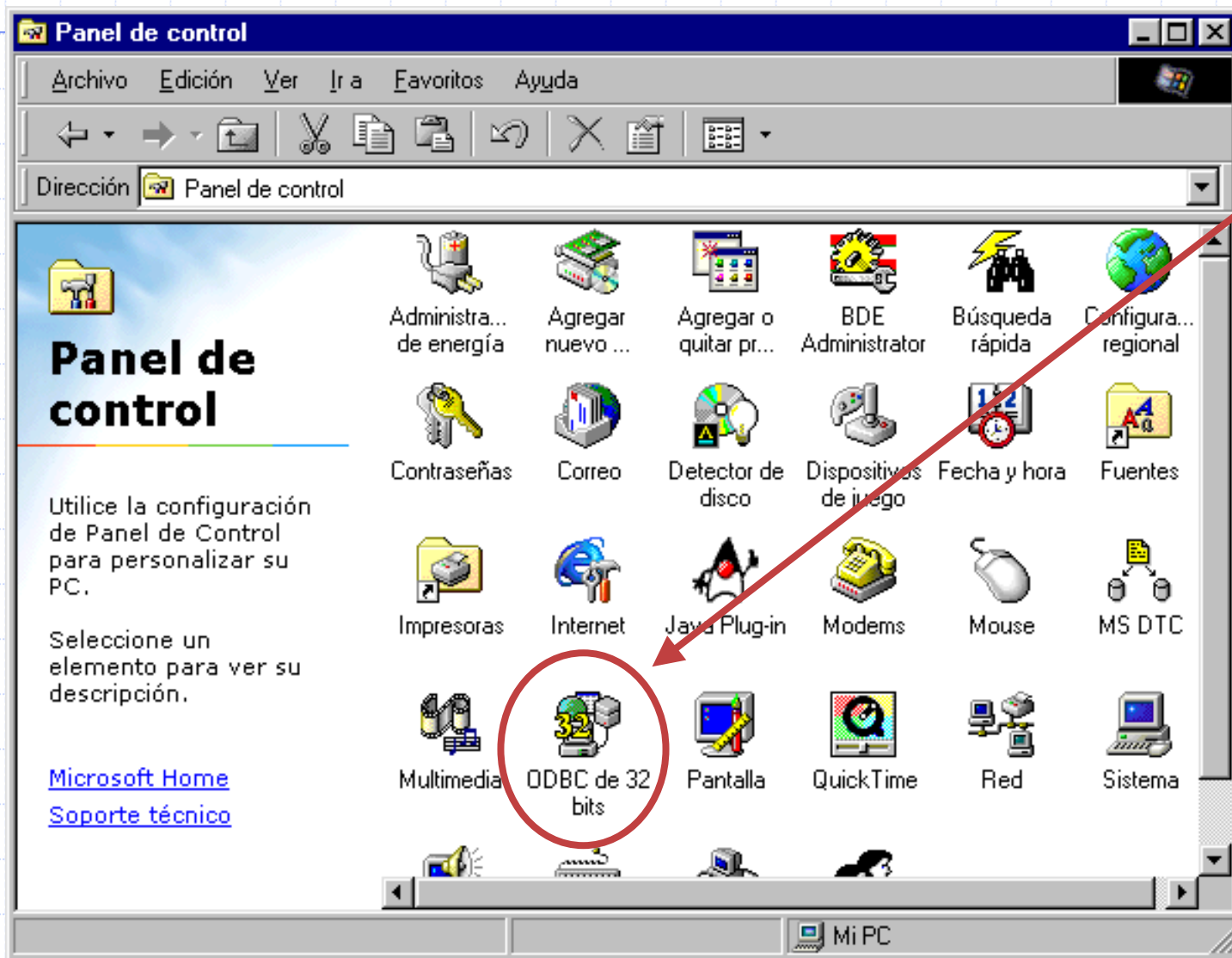
⊕ *Open DataBase Connectivity*



**ARQUITECTURA ODBC**

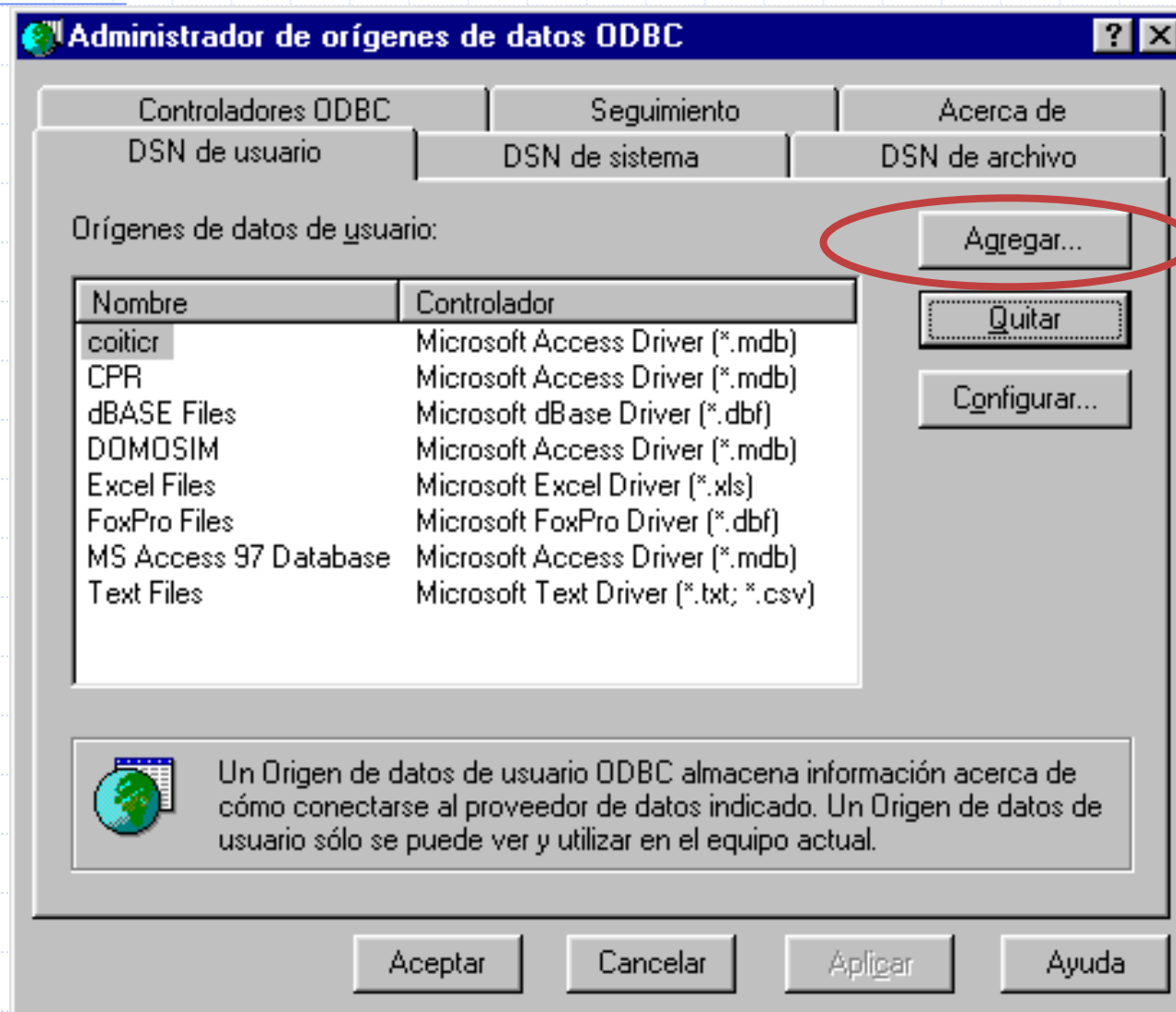
# El estándar JDBC

## Configuración JDBC:ODBC



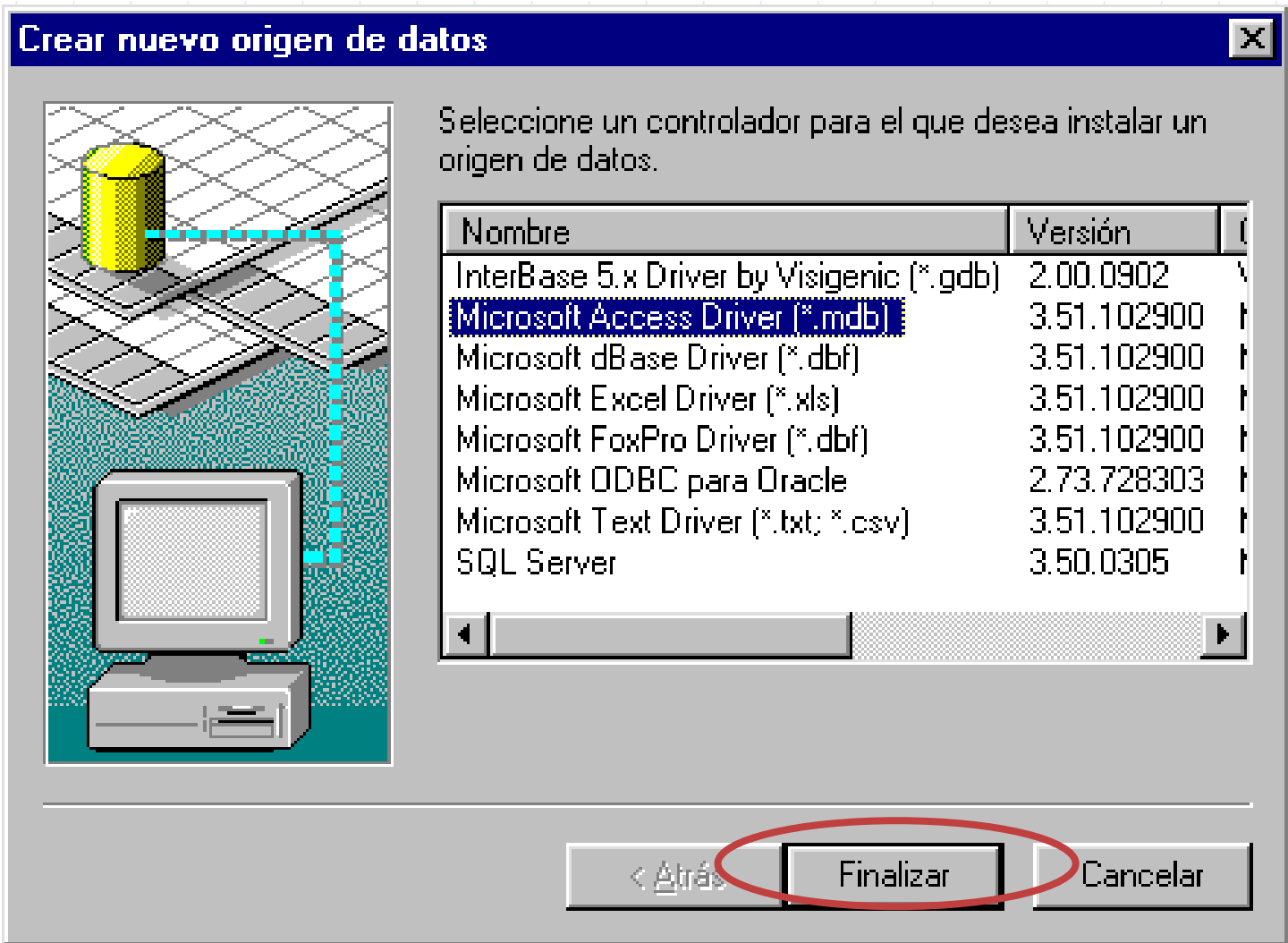
# El estándar JDBC

## Configuración JDBC:ODBC



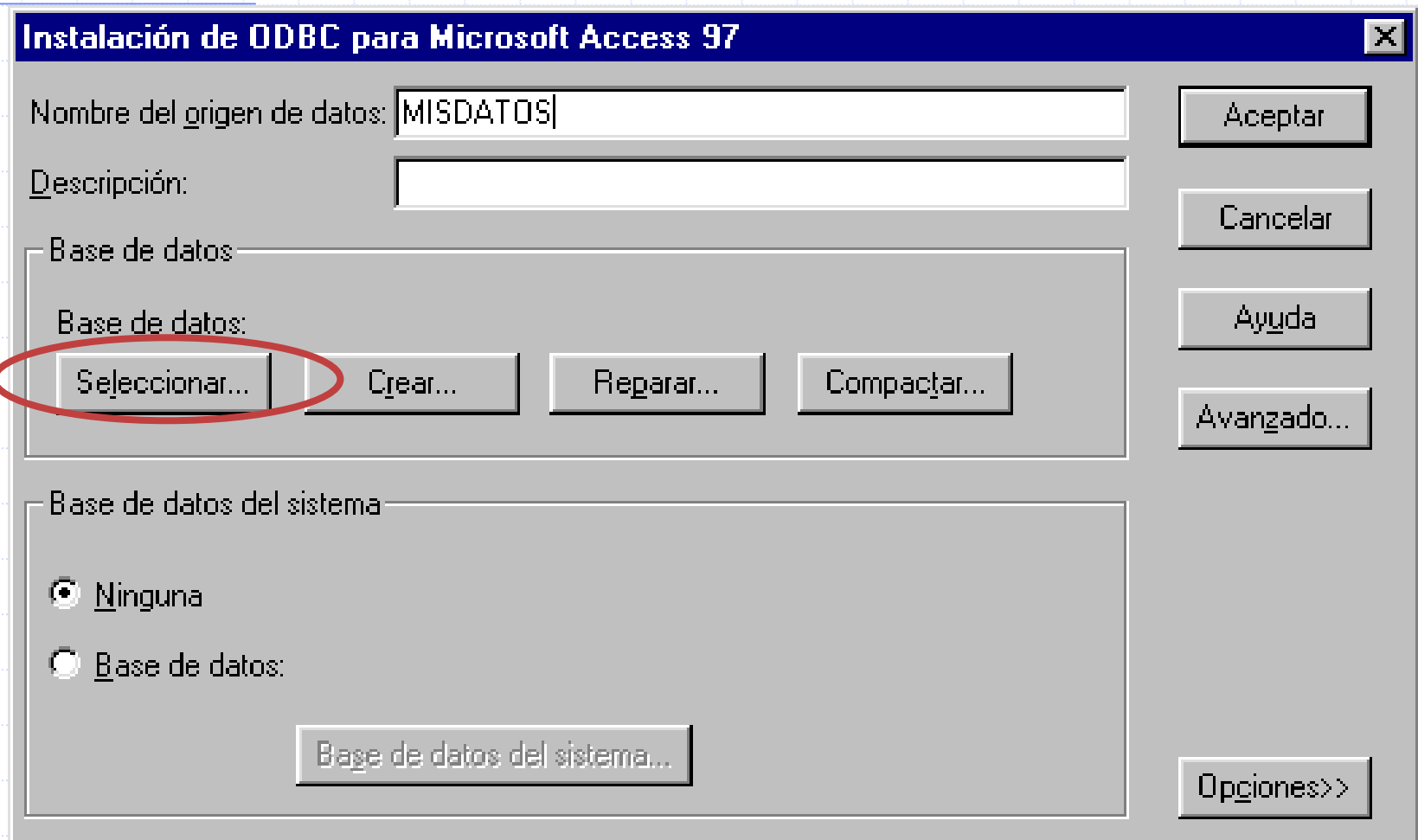
# El estándar JDBC

## Configuración JDBC:ODBC



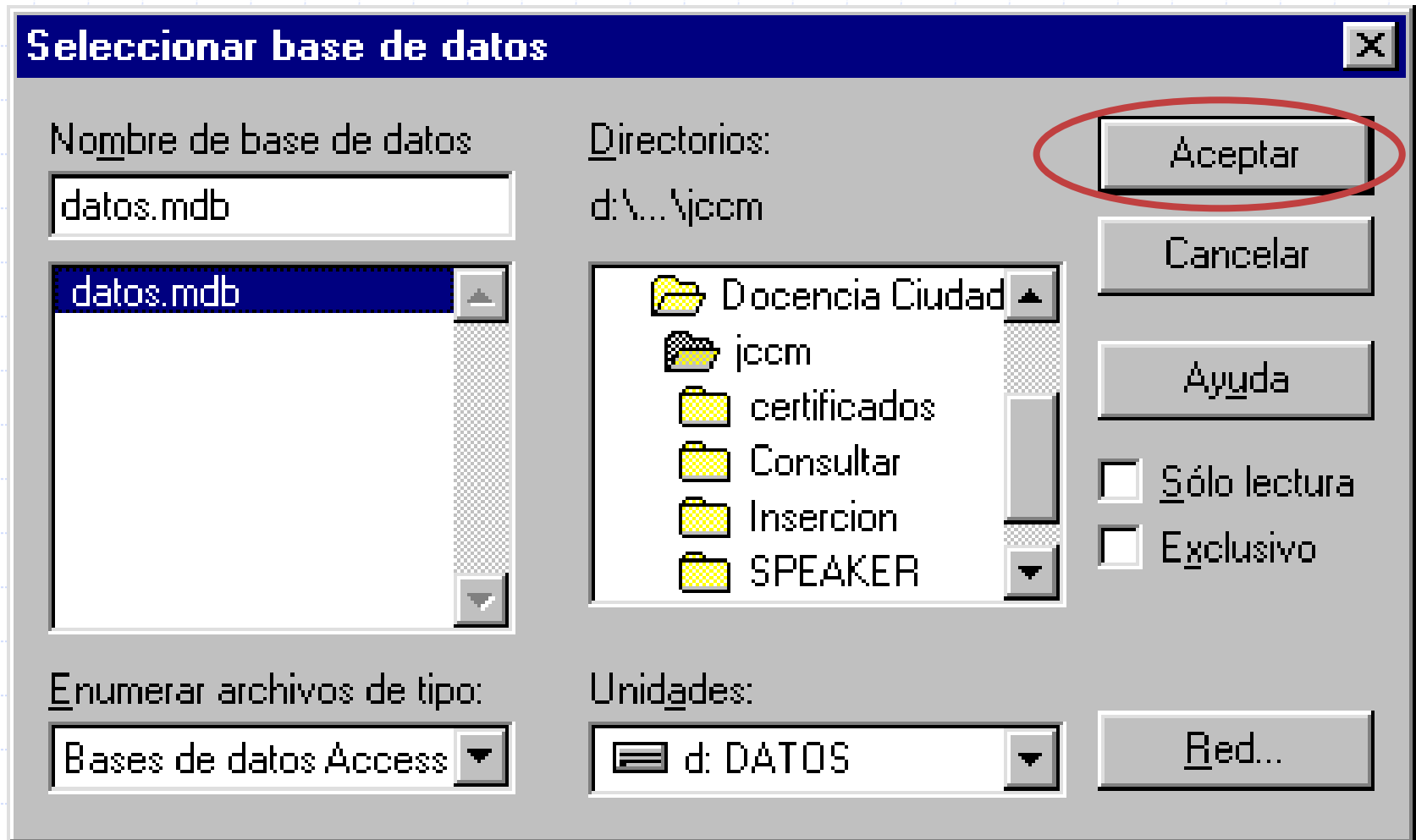
# El estándar JDBC

## Configuración JDBC:ODBC



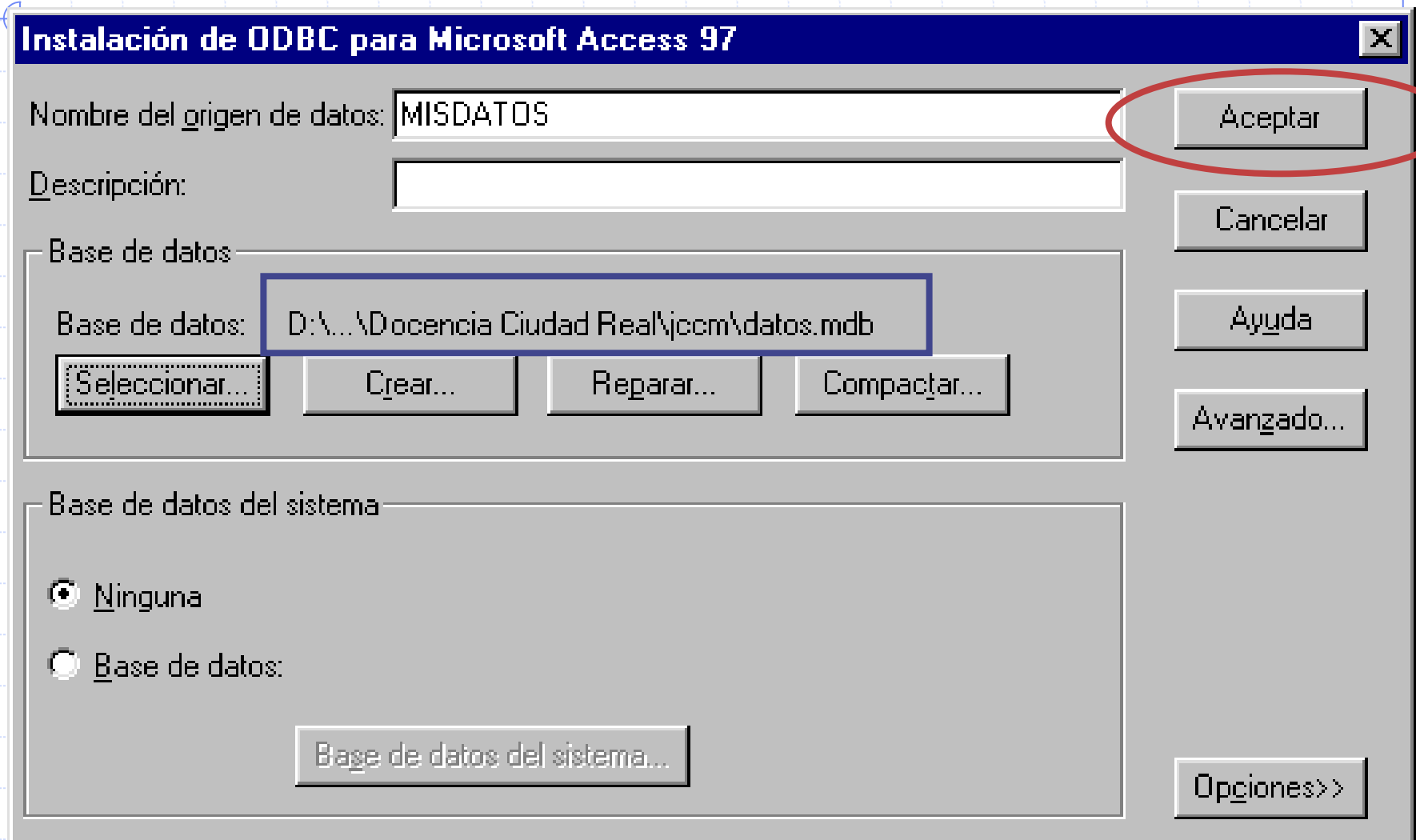
# El estándar JDBC

## Configuración JDBC:ODBC



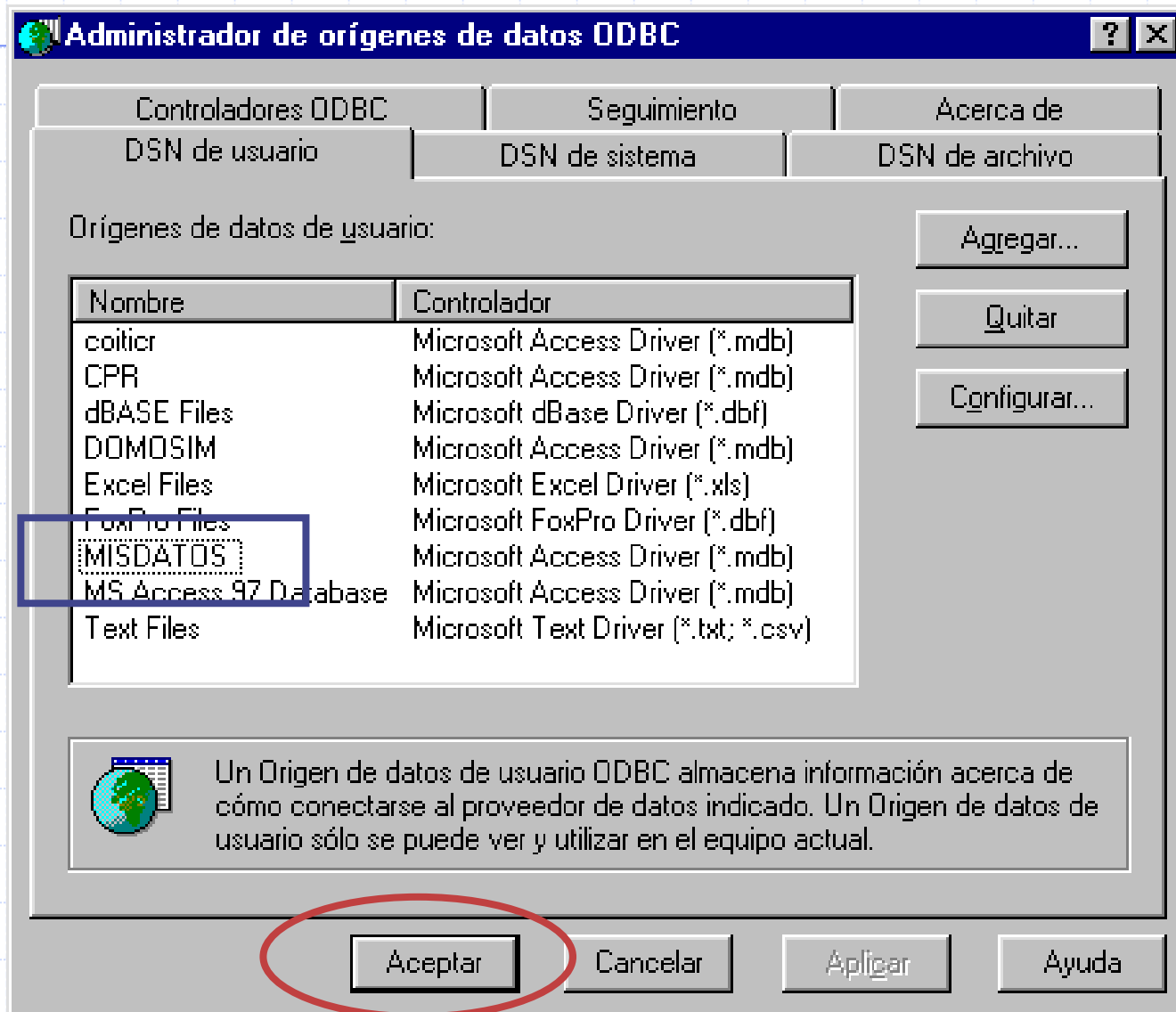
# El estándar JDBC

## Configuración JDBC:ODBC



# El estándar JDBC

## Configuración JDBC:ODBC



# El estándar JDBC

## Implementación

### ⊕ Paquete java.sql

- ⊕ Los programas deben declarar el uso de este paquete

```
import java.sql.*
```

### ⊕ El Gestor JDBC

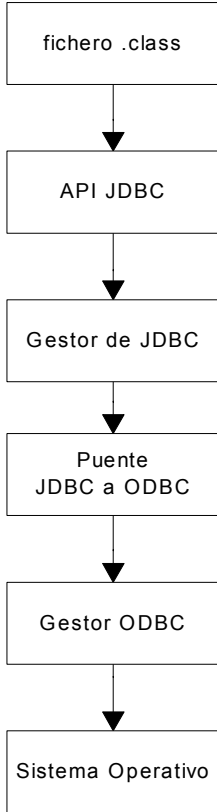
- ⊕ Para una base de datos concreta

- Oracle
- Borland

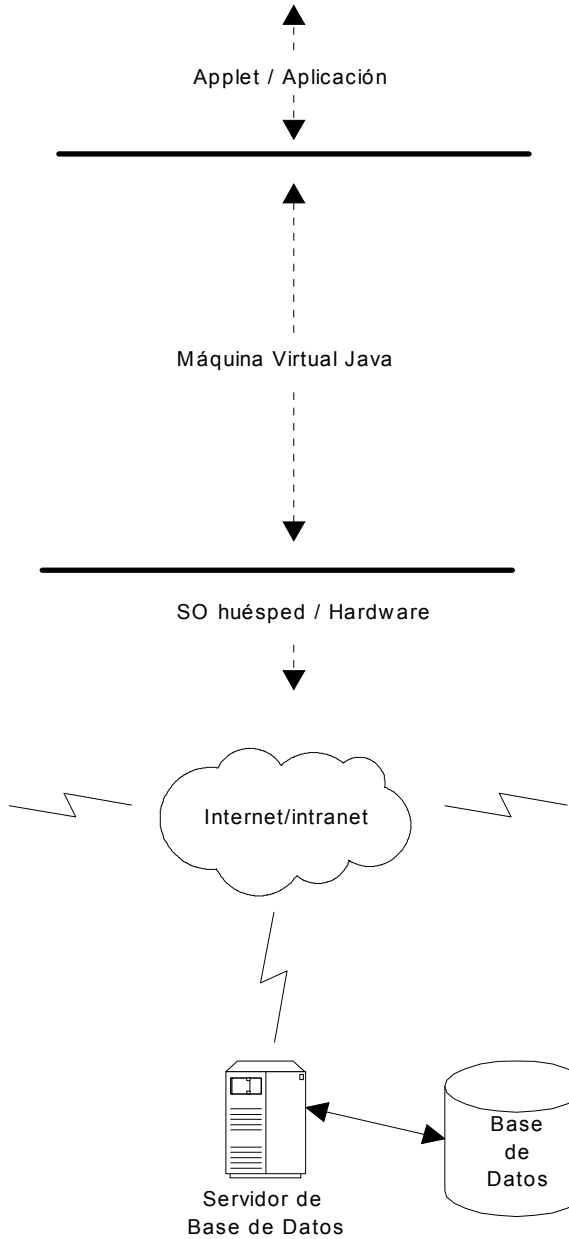
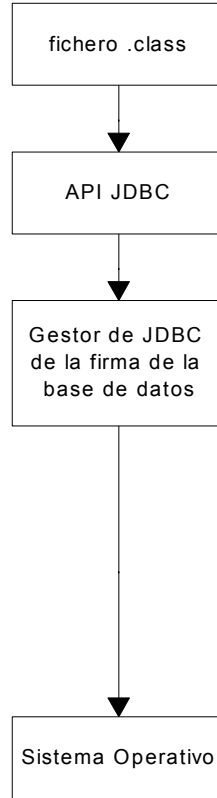
- ⊕ Genérico para varias bases de datos

- JDBC:ODBC
- Se traducen las llamadas JDBC en llamadas ODBC

Implementación de Sun del ODBC



Ejemplo de implementación de una firma de bases de datos





# El estándar JDBC

## Tipos de Clases

<b>TIPO</b>	<b>Clase JDBC</b>
<b>Implementación</b>	<b>java.sql.Driver</b> <b>java.sql.DriverManager</b> <b>java.sql.DriverPropertyInfo</b>
<b>Conexión a base de datos</b>	<b>java.sql.Connection</b>
<b>Sentencias SQL</b>	<b>java.sql.Statement</b> <b>java.sql.PreparedStatement</b> <b>java.sql.CallableStatement</b>
<b>Datos</b>	<b>java.sql.ResultSet</b>
<b>Errores</b>	<b>java.sql.SQLException</b> <b>java.sql.SQLWarning</b>

# Ejemplo con JDBC

## Consulta de datos

```
import java.sql.*;
```

```
.....
```

```
final String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
```

```
final String BBDD = "jdbc:odbc:ARTICULOS";
```

```
try {
```

```
    Class.forName(DRIVER);
```

```
    Connection conexion = DriverManager.getConnection(BBDD);
```

```
    PreparedStatement select = conexion.prepareStatement(  
        "SELECT * FROM articulos ORDER BY Titulo");
```

```
    ResultSet r = select.executeQuery();
```

```
    while (resultado.next()) {
```

```
        System.out.println(resultado.getString("Titulo"));
```

```
    }
```

```
    resultado.close(); select.close(); conexion.close();
```

```
    } catch (Exception e) {
```

```
        System.out.println("Error: " + e);
```

```
    }
```

Driver

Conexión

SQL

Acceso a  
los campos

# Ejemplo con JDBC

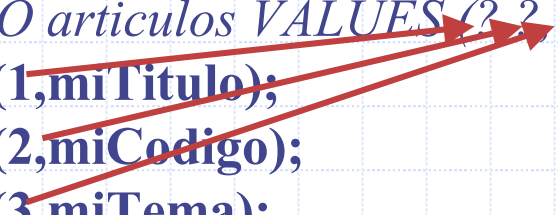
## Consulta de datos con condición

```
import java.sql.*;
final String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
final String BBDD = "jdbc:odbc:ARTICULOS";
try {
    Class.forName(DRIVER);
    Connection conexion = DriverManager.getConnection(BBDD);
    PreparedStatement select = conexion.prepareStatement(
        "SELECT * FROM articulos WHERE Titulo=?");
    select.setString(1, miTitulo.getText());
    ResultSet resultado = select.executeQuery();
    while (resultado.next()) {
        lista.addItem(resultado.getString("Titulo"));
    }
    resultado.close(); select.close(); conexion.close();
} catch (Exception e) {
    System.out.println("Error: " + e);
}
```

# Ejemplo con JDBC

## Inserción de datos

```
import java.sql.*;
final String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
final String BBDD = "jdbc:odbc:ARTICULOS";
try {
    Class.forName(DRIVER);
    Connection conexion = DriverManager.getConnection(BBDD);
    PreparedStatement insert = conexion.prepareStatement(
        "INSERT INTO articulos VALUES (? ? ?)");
    insert.setString(1,miTitulo);
    insert.setString(2,miCodigo);
    insert.setString(3,miTema);
    int resultado = insert.executeUpdate();
    insert.close(); conexion.close();
} catch (Exception e) {
    System.out.println("Error: " + e);
}
```



# Clases de JDBC

## java.sql.DriverManager

- ⊕ **Lleva el control de los gestores JDBC disponibles**
  - ⊕ Es posible que existan varios dentro del sistema
  - ⊕ Por defecto, carga todos los disponibles en *sql.drivers*
  - ⊕ El gestor cargado debería registrarse con el método *registerDriver*
- ⊕ **Sintaxis utilizada: URL's**
  - ⊕ jdbc:<subprotocolo>:<parámetros>
  - ⊕ jdbc:odbc:NOTICIAS:UID= Sistema;PWD= SistemaPW
- ⊕ **Seguridad**
  - ⊕ Hay que tener presente el modelo de seguridad

```
final String BBDD = "jdbc:odbc:ARTICULOS";  
Connection conexion =  
    DriverManager.getConnection(BBDD);
```

# Clases de JDBC

## java.sql.Driver

- ⊕ **Gestor de información y configuración general**
- ⊕ **Se carga durante la inicialización**
  - ⊕ **mediante**
    - ⊕ **DriverManager.registerDriver**
    - ⊕ **Class.forName**
- ⊕ **Se le pedirá información a lo largo del programa**
- ⊕ **Residirá en memoria**
- ⊕ **Métodos:**
  - ⊕ **connect**
  - ⊕ **getPropertyInfo**

```
final String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";  
Class.forName(DRIVER);
```

# Clases de JDBC

## java.sql.Connection

- ⊕ **Puntero a la base de datos**
- ⊕ **Proporciona el contexto de trabajo para los objetos Statement y ResultSet**
- ⊕ **Soporta propiedades de transacción**
  - ⊕ **setAutoCommit**
  - ⊕ **commit**
  - ⊕ **rollback**

```
Connection conexion =  
    DriverManager.getConnection(BBDD);
```

# Clases de JDBC

## java.sql.Statement

### ⊕ Ejecución de una sentencia SQL

#### ⊕ executeQuery

- ⊕ Sentencias SELECT

- ⊕ devuelve un ResultSet

#### ⊕ executeUpdate

- ⊕ Sentencias INSERT, DELETE, UPDATE, CREATE

- ⊕ Devuelve un entero

#### ⊕ execute

- ⊕ Sentencias desconocidas en tiempo de compilación o sentencias que devuelven resultados complejos

- ⊕ *Devuelve true/false*

```
Statement select = conexion.createStatement();
```

```
ResultSet resultado =
```

```
    select.executeQuery("SELECT * FROM ACTIVIDAD");
```

# Clases de JDBC

## java.sql.PreparedStatement

- ⊕ **Extiende Statement para añadir sentencias precompiladas SQL**
  - ⊕ **Compila la sentencia SQL la primera vez**
  - ⊕ **Sentencias que son llamadas más de una vez en el programa**
- ⊕ **Soporta parámetros de entrada**
  - ⊕ **setInt, setFloat, setLong, setString**

```
PreparedStatement select = conexion.prepareStatement(  
    "SELECT * FROM articulos WHERE Titulo=?");  
select.setString(1, "Mi titulo");  
ResultSet resultado = select.executeQuery();
```

# Clases de JDBC

## java.sql.ResultSet

- ⊕ **Contiene los datos resultado de una sentencia SQL**
- ⊕ **Se recuperan secuencialmente en filas**
  - ⊕ **next** sirve para avanzar una fila
- ⊕ **Se puede acceder a los datos de las columnas en cualquier orden**
  - ⊕ **índice de posición**
  - ⊕ **nombre del campo**
  - ⊕ **Métodos: getString, getFloat, getInt, etc.**
  - ⊕ **Método wasNull()**

```
PreparedStatement select = conexion.prepareStatement (
    "SELECT * FROM articulos WHERE Titulo=?");
select.setString(1,"Mi titulo");
ResultSet resultado = select.executeQuery();
while (resultado.next())
    System.out.println(resultado.getString("Titulo"));
```

# Ejemplo con JDBC

## Programa para trabajar con MySQL

```
import java.sql.*;
final String DRIVER = "org.gjt.mm.mysql.Driver";
final String BBDD = "jdbc:mysql://chico.inf-cr.uclm.es/articulos";
try {
    Class.forName(DRIVER);
    Connection conexion = DriverManager.getConnection(BBDD);
    Statement insert = conexion.prepareStatement(
        "INSERT INTO articulos VALUES (?, ?, ?)");
    insert.setString(1, miTitulo);
    insert.setString(2, miCodigo);
    insert.setString(3, miTema);
    int resultado = insert.executeUpdate();
    insert.close(); conexion.close();
} catch (Exception e) {
    System.out.println("Error: " + e);
}
```

# API JDBC 2.0

## Características

### ⊕ Requerimientos

- ⊕ Versión 1.2 o superior de JDK
- ⊕ Driver con soporte para JDBC 2.0
- ⊕ DBMS que implemente características de JDBC 2.0

### ⊕ Aportaciones

- ⊕ Ampliación de la interfaz ResultSet
  - ⊕ Movimiento a cualquier posición y dirección
- ⊕ Realización de modificaciones en los datos a través del ResultSet (sin SQL)
- ⊕ Soporte para ejecución Batch de sentencias
- ⊕ Nuevos tipos de datos correspondientes a SQL3