JDBC

JDBC

- Idea:dar a los programadores un API (Application Programming Interface) que les permita codificar de manera independiente al fabricante del gestor de base de datos
- JDBC tiene dos capas: por una parte el API JDBC y por otra el controlador o driver del fabricante

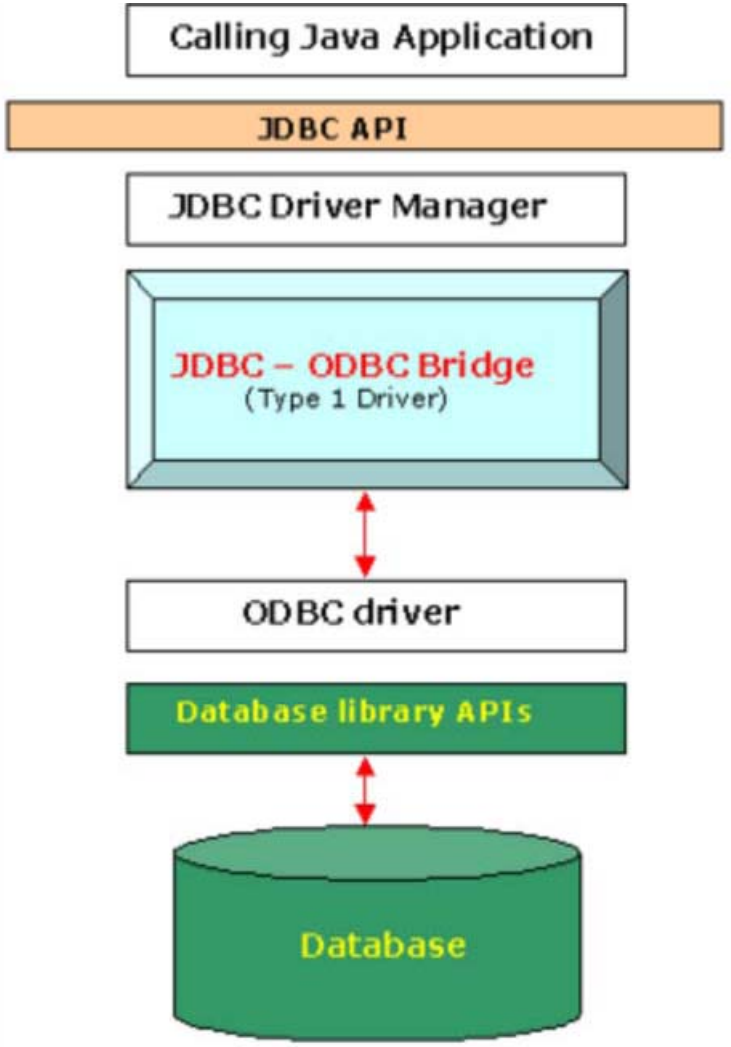
API JDBC

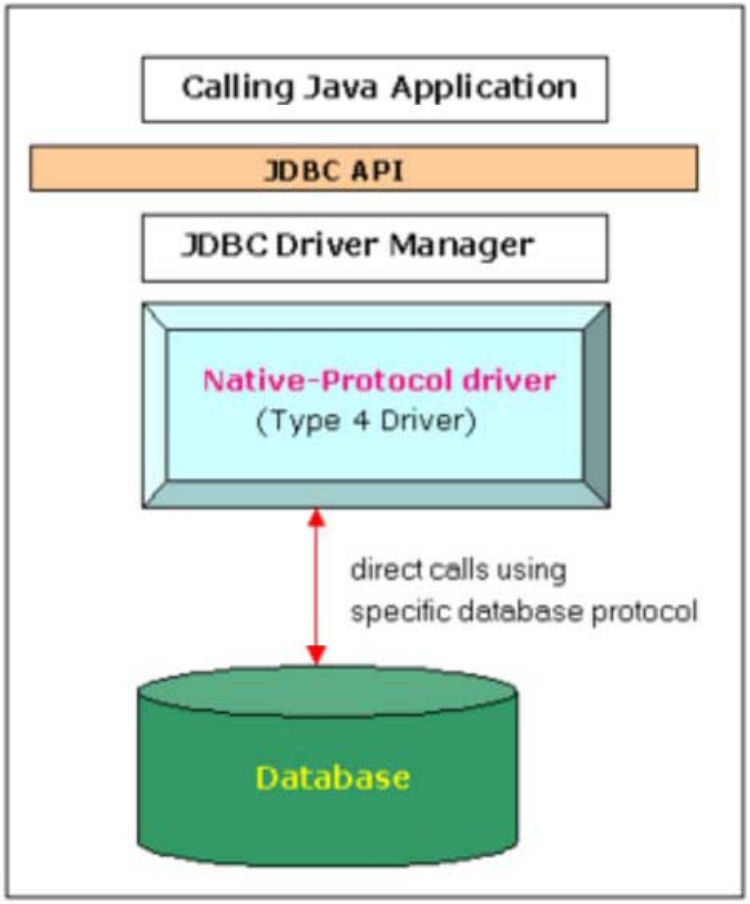
- Nos ofrece un conjunto de clases,métodos e interfaces que nos permite aislar el código de la aplicación de las características particulares de cada fuente de datos.

Driver

- El controlador (Driver) es un componente software que permite a las aplicaciones Java interactuar con bases de datos. Recibe las ordenes y las traduce a ordenes para ek gestor de base de datos.

Driver tipo 1 y tipo 4

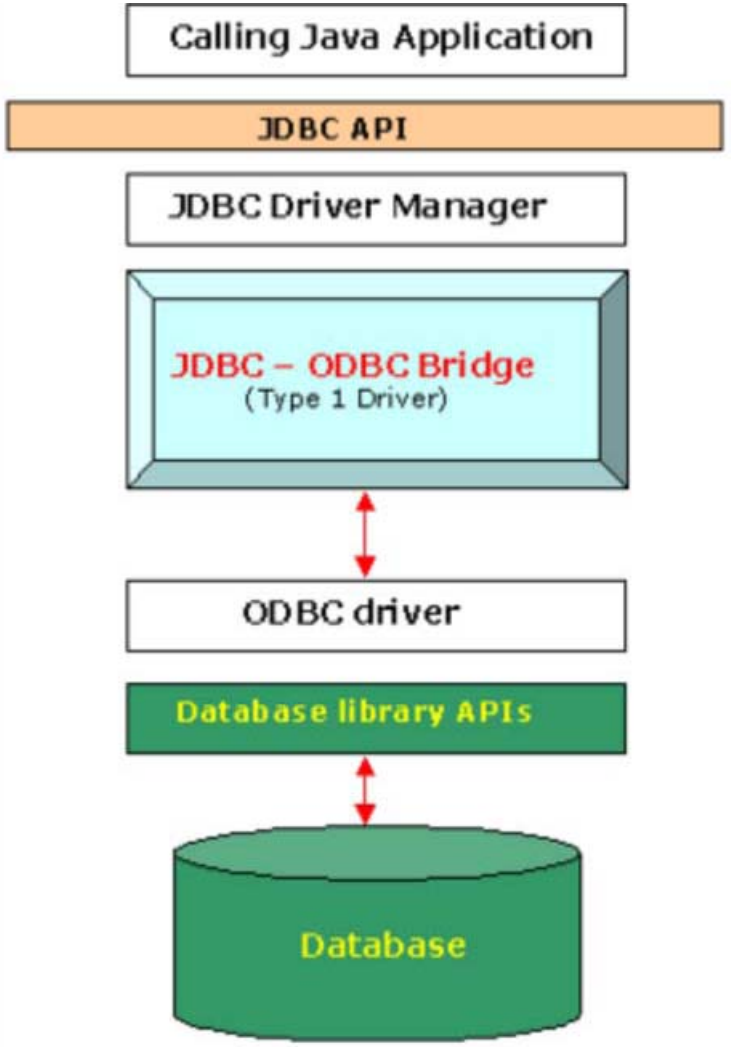


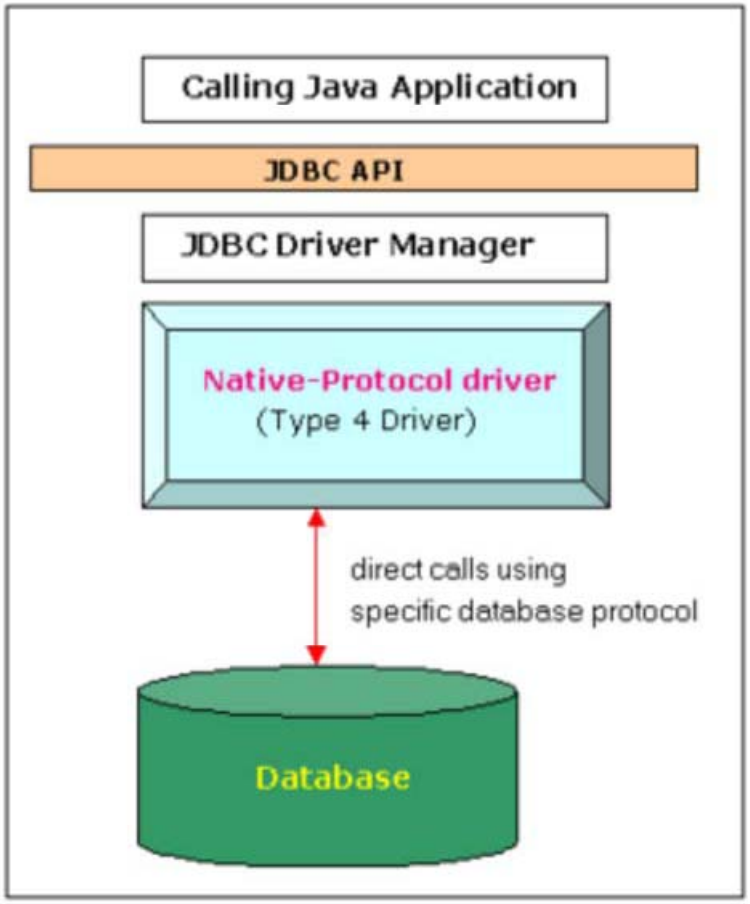


Driver tipo1

- Controlador que traduce JDBC a ODBC.
- Exige la instalación y configuración de ODBC en la computadora cliente
- El más lento de todos
- Sun recomienda usarlo solo para prototipado de aplicaciones

Driver tipo 1 y tipo 4





Driver tipo4

- El único que es Java 100%. Es el único que se comunica directamente con la BD y por ello es el más eficiente de todos los driver. Se puede usar en applets (por no usar código nativo). El más rápido de todos
- Cada SGBD requiere su Driver

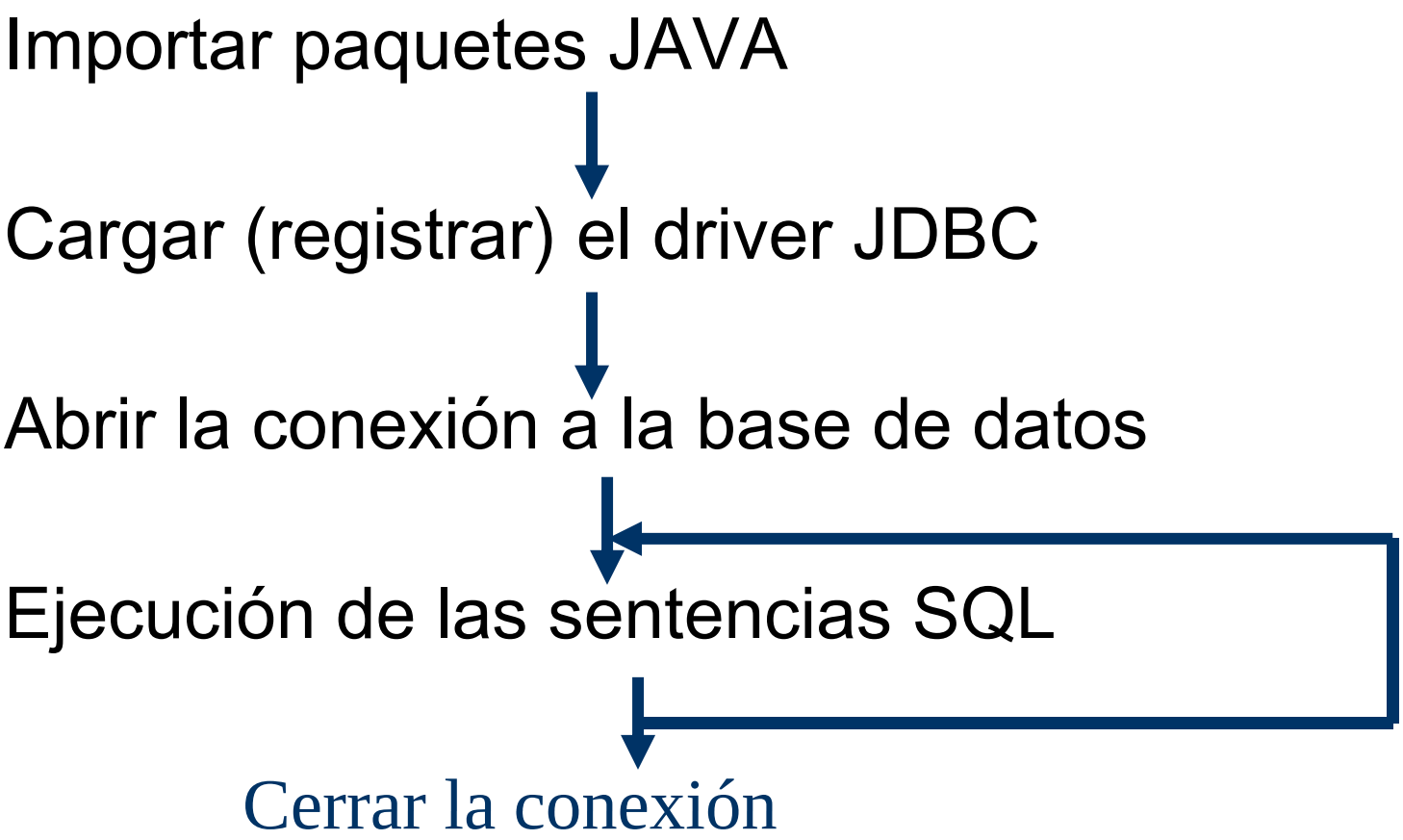
API JDBC

LSI-Ainhoa Álvarez

Principales clases e interfaces

Interfaces	**Descripción**
Driver	Localiza un driver para una BD específica
Connection	Conecta una aplicación a una base de datos
Statement	Container para una sentencia SQL
PreparedStatement	Precompila una sentencia SQL y luego se usa múltiples veces
CallableStatement	Ejecuta un procedimiento almacenado
Resultset	Filas devueltas cuando se ejecuta una query
ResultSetMetaData	Número, tipos y propiedades del resultSet
Clases	
DriverManager	Carga objetos driver y crea conexiones a BD
DriverPropertyInfo	Usado por clientes especializados

Pasos a seguir



Pasos 1 y 2

- **Importar paquetes** : import java.sql.*;
- **Registrar driver**:
  - Class.forName(String nombre_driver)

EJEMPLO

```
import java.sql.*;
public class Empresa1{
   public static void main (String args[]){
  //cargar el driver tipo1
try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  }catch (ClassNotFoundException e1){};

 //cargar el driver tipo4
try {
      Class.forName(“con.mysql.jdbc.Driver");
  }catch (ClassNotFoundException e1){};
```

Abrir y cerrar conexión

Abrir conexión, en la Clase DriverManager	
static connection	***getConnection**(String url).*
static connection	***getConnection(String** url, String user, String password)*

Cerrar conexión, en la Clase connection	
void	*close()*

Tipo 1

```
//Abrir la conexión a la base de datos
 Connection con;
     try{
   String url="jdbc:odbc:FuenteBodega";

   con=DriverManager.getConnection(url);

  System.out.println("conexion abierta");
```

Tipo 4

```
//Abrir la conexión a la base de datos
 Connection con;
     try{
    String  url
    ="jdbc:mysql://localhost:3306/bodegas?user=root&
    password=vppy108";
     con=DriverManager.getConnection(url);

  System.out.println("conexion abierta");
```

Ejecución sentencias

Ejecución de sentencias, en statement	
ResultSet	executeQuery()
int	executeUpdate() Returns either the row count for INSERT, UPDATE or DELETE statements, or 0 for SQL statements that return nothing

Ejecución sentencias, Statement

- Creación de statement: sobre la clase connection

Method Summary connection	
Statement	createStatement()

```
//generación del Statement
Statement st= con.createStatement();
//Ejecución
 int i=st.executeUpdate(“UPDATE empresa.empleado
    set nombre=“nombre1” where nombre=“nom2””);

ResultSet rs=st.executeQuery("SELECT * from
    empleado");
```

Resultset

Method Summary	
boolean	first()
boolean	getBoolean(int columnIndex)
Boolean	getBoolean(String colName)
byte	getByte(int columnIndex)
boolean	last()
Boolean	next()
Boolean	previous()
ResultSetMetaData	getMetaData()

Ejemplo Resultset

```
//Mostramos el resultado al usuario
//Moves the cursor down one row from its current position. A
    //ResultSet cursor is initially positioned before the first
    //row; the first call to the method next makes the first row
    //the current row; the second call makes the second row the
    //current row, and so on.
//Cuando no haya más filas devolverá false
while (rs.next()){
    System.out.println (rs.getInt("NSS")+" "
    +rs.getString("Nombre"));
}
```

Ejecución sentencias,PS

Method Summary connection	
PreparedStatement	prepareStatement (String sql)

Prepared Statement

```
//generación del preparedStatement
String sql= "SELECT * from departamento
   where nd=?";
 PreparedStatement ps= con.prepareStatement(
   sql);
```

PreparedStatement

Method Summary	
void	setFloat(int parameterIndex, float x)
void	setInt(int parameterIndex, int x)
ResultSetMetaData	getMetaData()

Prepared Statement

```
//damos valores a los parámetros
ps.setInt(1,2);
//ejecutamos la sentencia SQL
ResultSet rs=ps.executeQuery();
```

ResultSetMetaData

Method Summary	
Int	getColumnCount()
String	getColumnName(int column)
Int	getColumnType(int column)
String	getColumnTypeName(int column)

Metadata

```
//análisis de la metadata
 ResultSetMetaData metadata=rs.getMetaData();
 int columns=metadata.getColumnCount();
  System.out.println("Columnas "+columns);

  for (int i=1;i<=columns;i++)
    System.out.println("Columna "+i
    +metadata.getColumnName(i)+"
    "+metadata.getColumnTypeName(i));
```

Cierre conexión

```
//Cerramos la conexión
 con.close();
   }catch(SQLException e){ });};
  }
}
```