

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

CENTRO REGIONAL DE PANAMÁ OESTE

LICENCIATURA EN DESARROLLO DE SOFTWARE

Integrantes

Josué Estrada 8-1004-1431

Mario Duque 8-1000-989

Edwin Mina 8-1014-83

Alberto Rangel 8-987-442

Gene Micewicz E-8-197254

Luis Williams 8-1001-1617

Nobel de Gracia 8-1004-1100

Profesor

Israel O. Deago Q.

Materia

Desarrollo de Software

Grupo

9SL-121

Semestre

III

Año

2023

Introduccion

¿Qué es JDBC? Es una biblioteca de clases que permite la conexión con Bases de Datos utilizando Java.

Permite realizar operaciones (consultas, actualizaciones, ...) sobre base de datos relacionales utilizando SQL (Structured Query Language).

¿Qué ventajas ofrece acceder a la base de datos utilizando JDBC y Java? Que la aplicación será independiente de la plataforma y que se puede mover la aplicación de un sistema gestor de bases de datos a otro (por ejemplo de Oracle a MySQL o Microsoft SQL Server)

Investigación

1. Definición de ODBC

ODBC es una especificación de una interfaz de programación de aplicaciones interfaz de programación de aplicaciones (API) que permite a las aplicaciones acceder a varios sistemas de gestión de bases de datos utilizando el lenguaje de consulta estructurado (SQL).

ODBC permite una interoperabilidad máxima: una única aplicación puede acceder a varios sistemas de gestión de bases de datos. Esto le permite desarrollar, compilar e incluir una aplicación sin especificar como destino un tipo específico de origen de datos. Después los usuarios pueden añadir controladores de base de datos, que enlazan la aplicación con el sistema de gestión de bases de datos que elijan.

ODBC fue creado por el SQL Access Group y fue lanzado por primera vez en 1992

2. Diferencias entre ODBC Y JDBC

El punto que diferencia fundamentalmente a JDBC y ODBC es que:

- JDBC es dependiente del lenguaje y es específico de Java, mientras que el
- ODBC es un lenguaje independiente.

Veamos en cuántos aspectos el JDBC y el ODBC difieren entre sí con la ayuda del cuadro de comparación que se muestra a continuación.

Bases para la comparación	JDBC	ODBC
BASIC	JDBC depende del lenguaje y la plataforma (específico de Java).	ODBC es lenguaje y plataforma independiente.
Forma completa	Conectividad de base de datos Java.	Conectividad abierta de la base de datos.
Código	El código es fácil de entender.	El código es complejo.

Definición de JDBC

Es una interfaz estándar entre cualquier aplicación Java y diferentes bases de datos. La función de JDBC es ayudar a la aplicación basada en Java a acceder a diferentes tipos de bases de datos. JDBC proporciona métodos para consultar la base de datos, y también se puede utilizar para actualizar la base de datos. JDBC proporciona **controladores JDBC** que convierten la solicitud de la aplicación Java en el lado del cliente al idioma que la base de datos entiende.

Definición de ODBC

ODBC es **Conectividad abierta de bases de datos**, ayuda a una aplicación a acceder a los datos de la base de datos. Una aplicación escrita en cualquier idioma puede usar ODBC para acceder a diferentes tipos de bases de datos y, por lo tanto, se dice que es independiente del idioma y la plataforma. Al igual que JDBC, ODBC también proporciona **controladores ODBC** que convierten la solicitud de una aplicación escrita en cualquier idioma en un lenguaje comprensible para las bases de datos.

Diferencias clave entre JDBC y ODBC

1. La diferencia más básica entre JDBC y ODBC es que JDBC depende del idioma y de la plataforma. Por otro lado, el ODBC es dependiente del lenguaje y la plataforma.
2. Java Database Connectivity es un acrónimo de JDBC y, por otro lado, Open Database Connectivity es un acrónimo de ODBC.
3. El código para ODBC es complejo y es difícil de aprender. Sin embargo, el código para JDBC es más simple y fácil de ejecutar.

Semejanza:

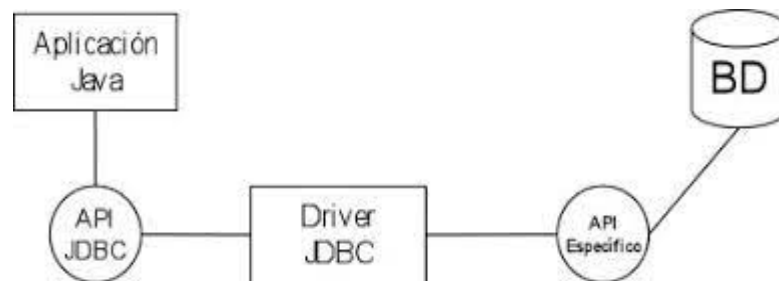
Las aplicaciones del lado del cliente utilizan ambas para acceder a diferentes tipos de bases de datos en el lado del servidor.

3. Explicación sobre el Puente JDBC-ODBC

El puente JDBC-ODBC es una interfaz que permite que las aplicaciones Java accedan a bases de datos relacionales utilizando controladores ODBC. Proporciona una forma de interconectar la API Java JDBC con los controladores ODBC existentes, lo que permite que las aplicaciones Java interactúen con las bases de datos a través de ODBC.

El puente JDBC-ODBC es útil en situaciones en las que no hay un controlador JDBC específico disponible para una base de datos específica, pero existe un controlador ODBC correspondiente. En este escenario, el puente JDBC-ODBC actúa como puente entre la aplicación Java y la base de datos a través del controlador ODBC.

Cuando se utiliza el puente JDBC-ODBC, la aplicación Java se conecta al controlador JDBC-ODBC, que a su vez se conecta al controlador ODBC. La API de JDBC proporciona los métodos necesarios para enviar consultas SQL a una base de datos y recuperar resultados a través de un controlador ODBC.



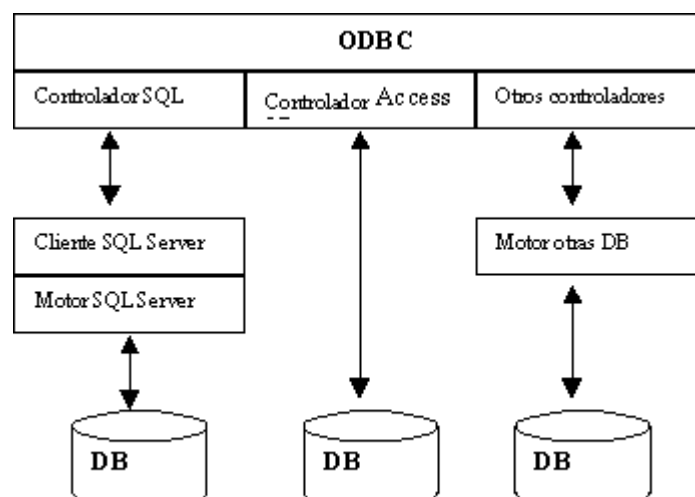
Sin embargo, cabe señalar que el puente JDBC-ODBC puede tener ciertas limitaciones en términos de rendimiento y funcionalidad. Dado que JDBC y ODBC son tecnologías diferentes, la transferencia de datos entre aplicaciones Java y bases de datos puede ser ineficiente. Además, es posible que algunas funciones avanzadas y funciones específicas del controlador JDBC no estén disponibles a través del puente JDBC-ODBC.

En otras palabras, JDBC-ODBC es una solución que permite que las aplicaciones de Java utilicen controladores ODBC existentes para acceder a bases de datos relacionales cuando los controladores JDBC específicos no están disponibles. Si bien puede ser una opción conveniente en algunas situaciones, es importante considerar sus limitaciones y evaluar si un controlador JDBC nativo es más adecuado para aprovechar todas las funciones de la base de datos.

4. Usos de ODBC

La utilización de ODBC permite crear aplicaciones de base de datos con acceso a cualquier base de datos en la que el usuario final tenga un controlador ODBC. ODBC proporciona una API que permite que la aplicación sea independiente del sistema de administración de bases de datos (DBMS) de origen.

Puede ayudar a ocultar las diferencias entre catálogos del sistema de servidores de bases de datos diferentes. le permite acceder a datos mediante llamadas de función ODBC en la aplicación. La interfaz ODBC elimina la necesidad de complicación previa y vinculación de la aplicación y aumenta la portabilidad de la aplicación.



5. Manual

Los pasos para usar ODBC son:

1. Configurar la interfaz ODBC, para ello el programa asigna en primer lugar un entorno SQL con la función `SQLAllocHandle()`, después un manejador para la conexión a la base de datos basada en el entorno anterior, el propósito de este manejador es traducir las consultas de datos de la aplicación en comandos que el sistema de base de datos entienda.
2. ODBC define varios tipos de manejadores:
 - `SQLHENV`: define el entorno de acceso a los datos
 - `SQLHDBC`: identifica el estado y configuración de la conexión
 - `SQLHDESC`: declaración SQL y cualquier conjunto de resultados asociados
 - `SQLHDESC`: recolección de metadatos utilizados para describir una sentencia SQL.
1. Una vez reservados los manejadores el programa abre la conexión a la base de datos usando `SQLDriverConnect()` o `SQLConnect()`.
2. Una vez realizada la conexión el programa puede enviar órdenes SQL a la base de datos usando `SQLExecDirect()`.
3. Al final de la sesión el programa se desconecta de la base de datos y libera la conexión y los manejadores del entorno SQL.

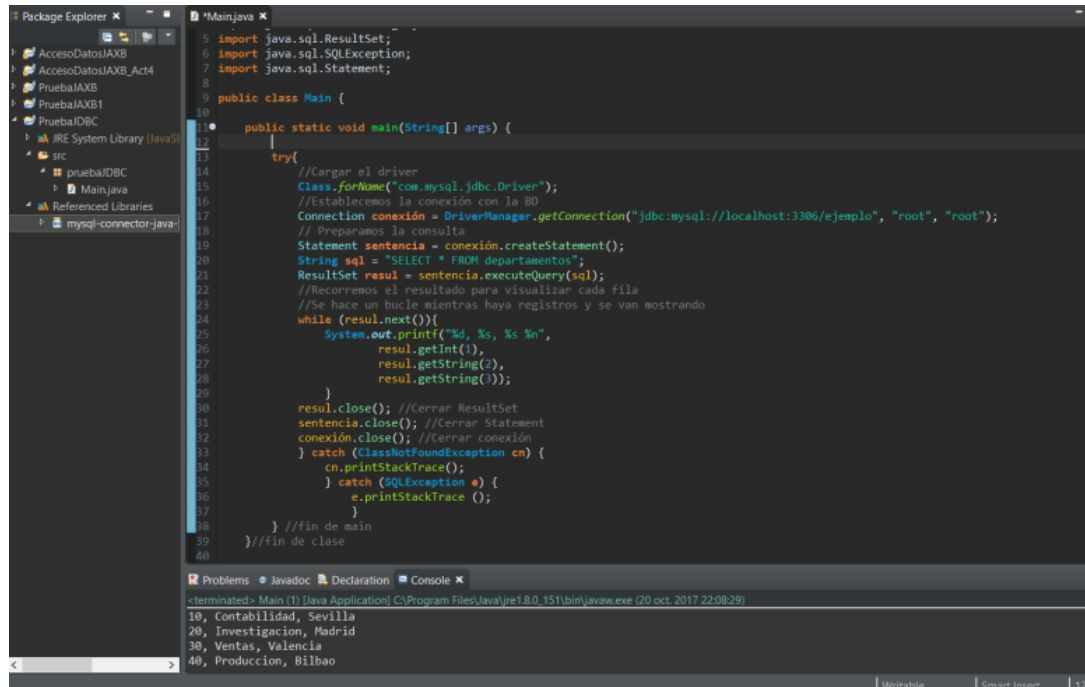
Para conectar JDBC con ODBC se usa:

JDBC-ODBC Bridge: permite el acceso a bases de datos JDBC mediante un driver ODBC. Convierte las llamadas al API de JDBC en llamadas ODBC. Exige la instalación y configuración de ODBC en la máquina cliente.

Ejemplo:

1. Primero creamos una base de datos y un usuario, la clave del usuario es la misma. Este usuario tendrá todos los privilegios sobre esta base de datos.
2. Después que tenemos la base de datos creamos las tablas con los nombres (empleados y departamentos) .
3. Luego de crear las tablas que queramos insertamos datos en las tablas.
4. Para poder probar el programa hemos de obtener el JAR que contiene el driver e incluirlo en el CLASSPATH o añadirlo a nuestro IDE.
5. También se ha incluido todo el programa en un try-catch ya que casi todos los métodos relativos a la base de datos pueden lanzar la excepción `SQLException`. La llamada al método `forName()` para cargar el driver puede lanzar la excepción `ClassNotFoundException` si este no se encuentra.

Anexos del Ejemplo:




The screenshot shows an IDE with a Package Explorer on the left and a Main.java file open. The code in Main.java is as follows:

```
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.sql.Statement;
8
9 public class Main {
10
11     public static void main(String[] args) {
12         try {
13             //Cargar el driver
14             Class.forName("com.mysql.jdbc.Driver");
15             //Establecemos la conexión con la BD
16             Connection conexión = DriverManager.getConnection("jdbc:mysql://localhost:3306/ejemplo", "root", "root");
17             // Preparamos la consulta
18             Statement sentencia = conexión.createStatement();
19             String sql = "SELECT * FROM departamentos";
20             ResultSet resul = sentencia.executeQuery(sql);
21             //Recorremos el resultado para visualizar cada fila
22             //Se hace un bucle mientras haya registros y se van mostrando
23             while (resul.next()){
24                 System.out.printf("%d, %s, %s\n",
25                     resul.getInt(1),
26                     resul.getString(2),
27                     resul.getString(3));
28             }
29             resul.close(); //Cerrar ResultSet
30             sentencia.close(); //Cerrar Statement
31             conexión.close(); //Cerrar conexión
32         } catch (ClassNotFoundException cn) {
33             cn.printStackTrace();
34         } catch (SQLException e) {
35             e.printStackTrace();
36         }
37     }
38 } //fin de main
39 } //fin de clase
40
```

The console output shows the result of the query:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (20 oct. 2017 22:08:29)
10, Contabilidad, Sevilla
20, Investigacion, Madrid
30, Ventas, Valencia
40, Produccion, Bilbao
```



The screenshot shows a SQL query editor with two queries. The first query is an INSERT statement for the 'departamentos' table, and the second query is an INSERT statement for the 'empleados' table.

```
1 • INSERT INTO departamentos(dept_no,dnombre,loc) VALUES
2 (10,"Contabilidad","Sevilla"),
3 (20,"Investigacion","Madrid"),
4 (30,"Ventas","Valencia"),
5 (40,"Produccion","Bilbao");
6
7 • INSERT INTO empleados(emp_no,apellido,dir,fecha_alt,salario,comision,dept_no) VALUES
8 (7369,"Sanchez","Empleado",7902,"1990-12-17",1040,NULL,20),
9 (7499,"Arroyo","Vendedor",7698,"1990-02-20",1500,390,30),
10 (7521,"Sala","Vendedor",7698,"1991-04-22",1625,650,30);
```

Conclusiones

-Gracias a estas especificaciones como bibliotecas de interfaces nos ayuda a que los programadores tanto de java como de otros lenguajes de programación pueden conectar su programa con la base de datos más fácil y con una especificación de por sí del mismo lenguaje y el uso de los programas pueda estar vinculados con bases de datos