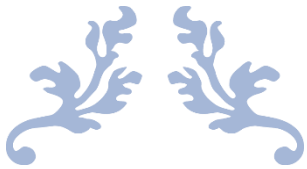


JDBC Y ODBC

Acceso a Datos

25 DE OCTUBRE DE 2023

CARLOS MORAIS BLANCO

Índice

Introducción

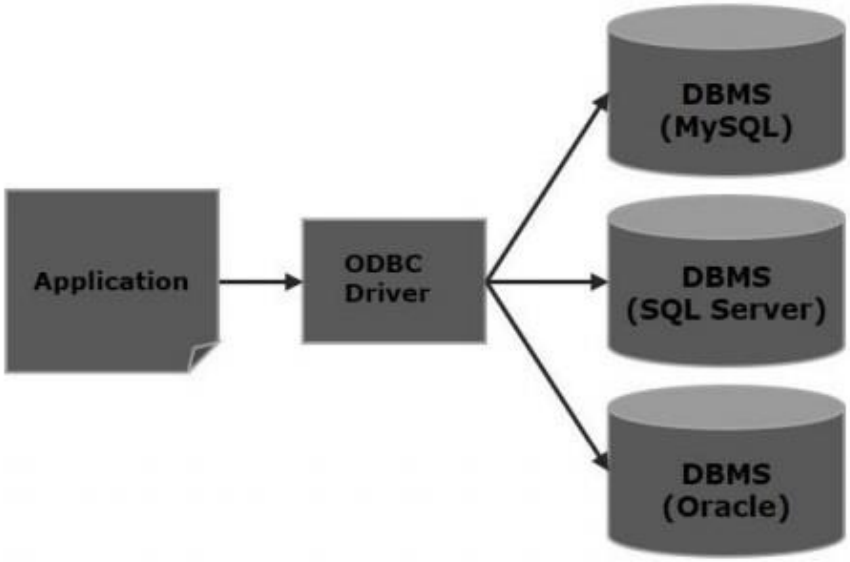
Este trabajo se centra en la documentación de la API JDBC (Java Database Connectivity), una herramienta fundamental en el desarrollo de aplicaciones Java que interactúan con sistemas de gestión de bases de datos. JDBC desempeña un papel crítico al actuar como el puente entre las aplicaciones Java y los datos almacenados en bases de datos.

En el transcurso de este trabajo, se revisarán los tipos de controladores JDBC, los modelos de conexión que ofrece, y examinaremos en detalle las clases clave en Java que facilitan la comunicación con bases de datos. Además, se definirá ODBC (Open Database Connectivity), un estándar influyente en el desarrollo de controladores JDBC.

ODBC

ODBC (Open database connectivity) es un sistema de controladores que actúan de intermediario entre las aplicaciones y las bases de datos. Se encargan de traducir las peticiones que se realizan en el código de la aplicación en comandos que utilicen las bases de datos, como pueden ser: Insert, Delete, Selects…

Es independiente de las bases de datos, SGBD y sistemas operativos. Una vez creada una aplicación que use ODBC, se puede usar en diferentes Sistemas gestores de bases de datos con poco cambio en la forma de acceder a los datos. ODBC usa lenguaje SQL y no puede usarse para bases de datos no relacionales. Esta desarrollado para el uso en aplicaciones escritas en C. Para utilizarlo, hace falta instalar un driver en el cliente que ejecuta la aplicación.



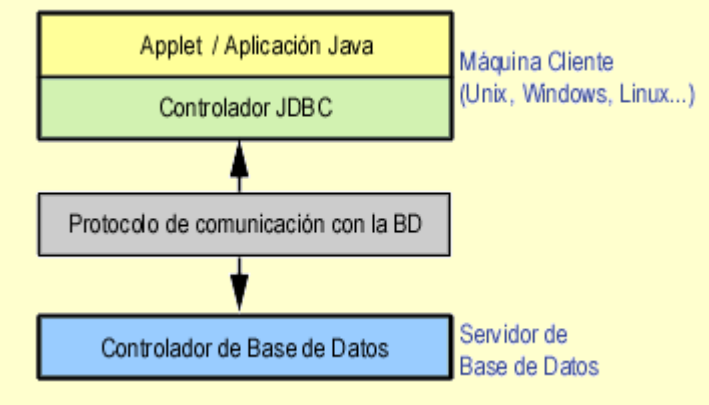
JDBC

JDBC es básicamente lo mismo que ODBC, pero para el uso en aplicaciones con lenguaje Java. El principal motivo de su creación, aunque ya esté JDBC, es que al ser C un lenguaje estructurado, si se usase ODBC para aplicaciones java, estas perderían las ventajas de la orientación a objetos, como puede ser su portabilidad. A mayores, existe un tipo de driver de JDBC, que permite su uso sin la necesidad de su instalación en el cliente.

Modelos de acceso a datos con JDBC

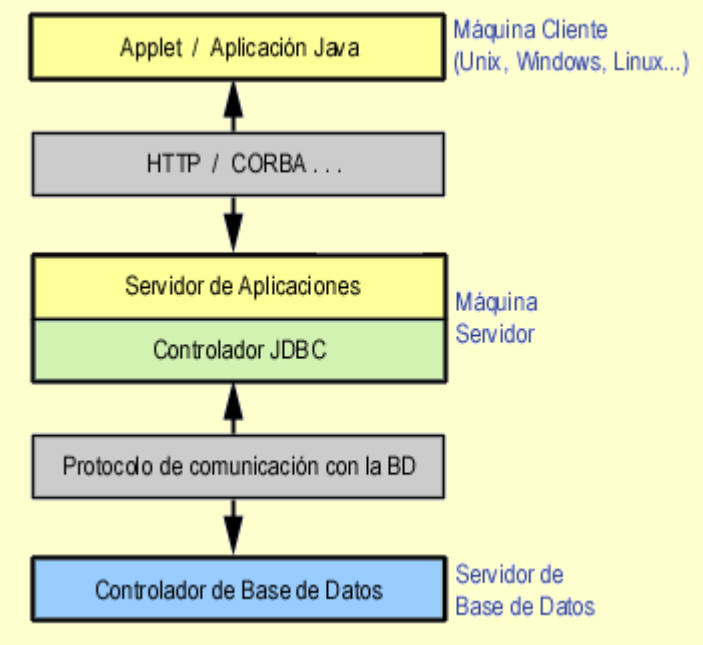
Modelo de dos capas

Este modelo se basa en la conexión de la aplicación Java y la base de datos directamente a través del driver JDBC.



Modelo de tres capas

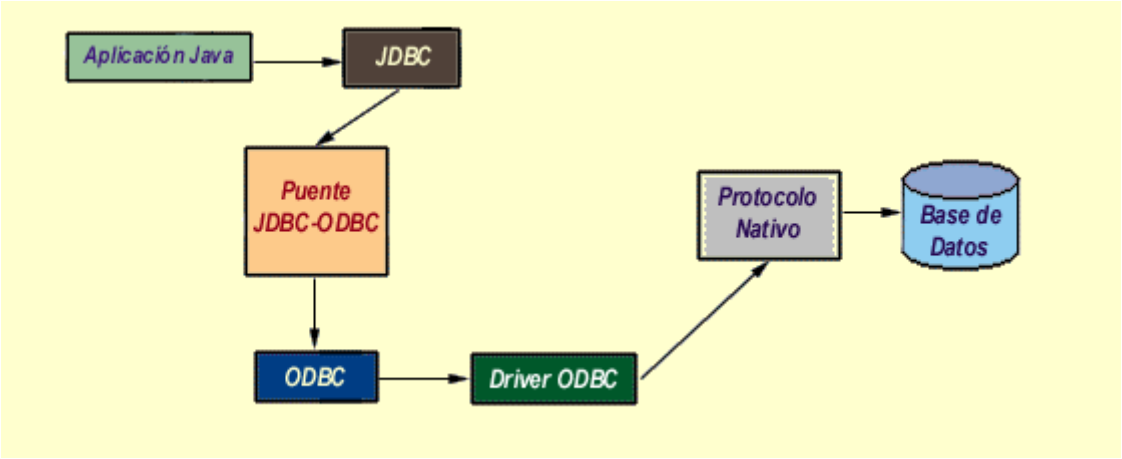
En este modelo, las instrucciones son enviadas a una capa intermedia entre el cliente y el servidor, que es la que se encarga de enviar las sentencias SQL a la base de datos y recoger el resultado desde la base de datos. En este caso el usuario no tiene contacto directo, ni a través de la red, con la máquina donde reside la base de datos.



Tipos de driver JDBC

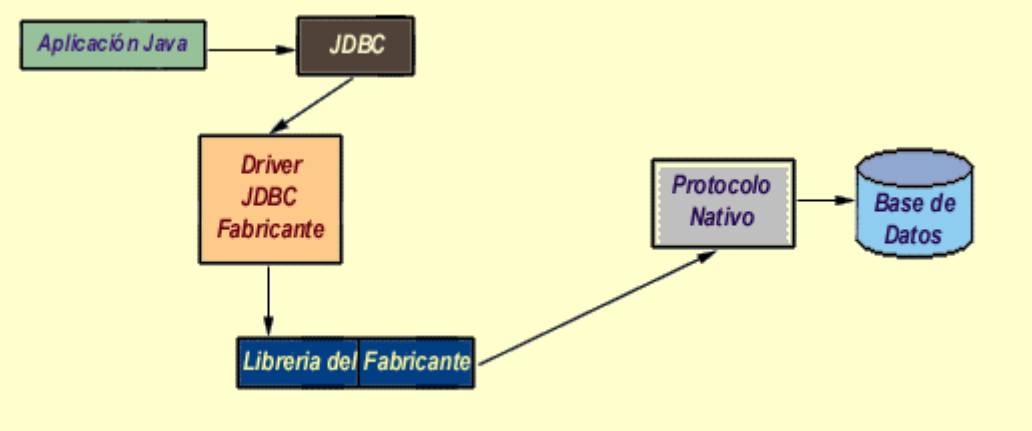
Puente JDBC-ODBC

Fue el primer driver creado, implementa operaciones JDBC traduciéndolas a operaciones ODBC. Se debe señalar que en cada máquina cliente que utilice el driver es necesaria una configuración previa, deberemos definir la fuente de datos utilizando para ello el gestor de drivers ODBC. Debido a esta configuración en las máquinas clientes, este tipo de driver no es adecuado para utilizarlo dentro de applets, su utilización está más encaminada a aplicaciones Java dentro de pequeñas intranets. Además, debido a que el puente JDBC-ODBC contiene parte de código nativo, no es posible utilizarlos dentro de applets, debido a las restricciones de seguridad de éstos.



Nativo

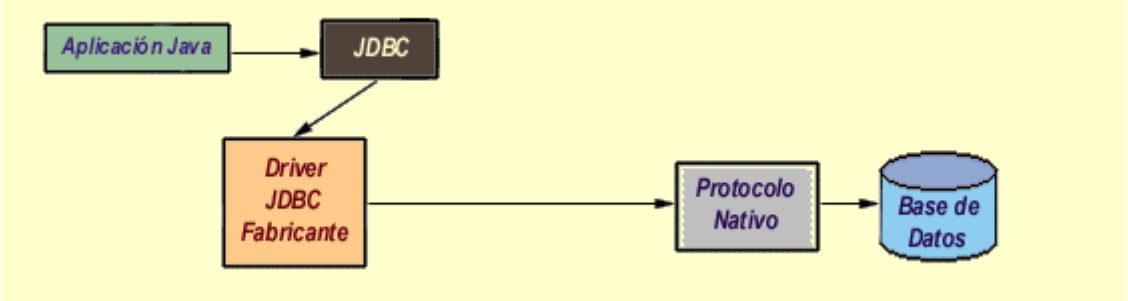
Este driver se salta la capa ODBC y se comunica directamente con la librería del SGBD. Este driver utiliza únicamente java, pero necesita la existencia de código binario en la máquina del cliente.



En red

Se comunica con el SGBD utilizando el protocolo de red nativo del servidor. Por consecuencia, el driver no necesita intermediario con el servidor. Así, este driver es independiente de la máquina en la que se va a ejecutar el programa. Dependiendo de la forma en que esté programado el driver, puede no necesitar ninguna clase de configuración por parte del usuario.

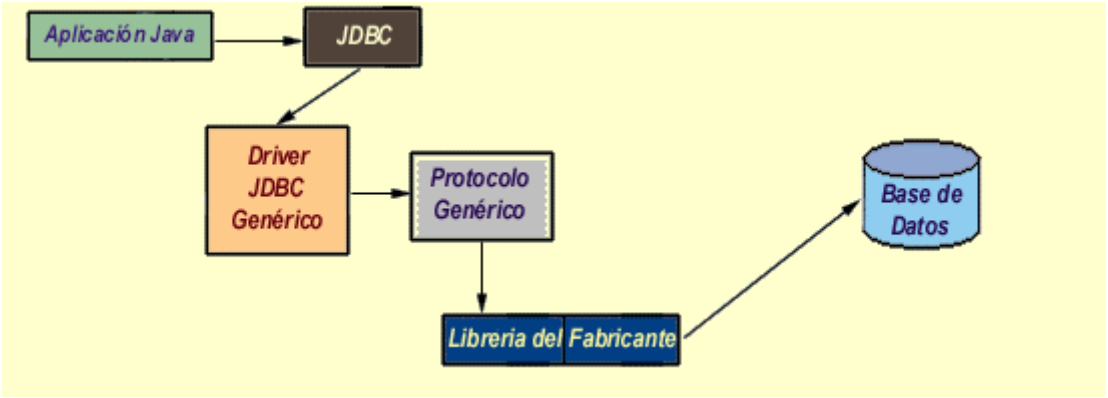
La única desventaja de este tipo de drivers es que el cliente está ligado a un SGBD concreto, ya que el protocolo de red que utiliza MySQL Workbench por ejemplo no tiene nada que ver con el utilizado por DB2 u Oracle. Usados en modelos de tres capas.



Thin

Requiere la presencia de un intermediario en el servidor. El driver JDBC hace las peticiones de datos al intermediario en un protocolo de red independiente del SGBD. El intermediario a su vez, que está ubicado en el lado del servidor, convierte las peticiones JDBC en peticiones nativas del SGBD.

La ventaja de este método es: el programa que se ejecuta en el cliente, y aparte de las ventajas de los drivers 100% Java, también presenta la independencia respecto al sistema de bases de datos que se encuentra en el servidor.



Clases e interfaces que intervienen

Driver

Permite a Java cargar y registrar controladores de bases de datos.

DriverManager

Se encarga de gestionar todos los drivers que puede utilizar una aplicación para las diferentes conexiones a bases de datos.

DriverPropertyInfo

Proporciona información a cerca de un Driver.

Connection

Representa una conexión física con la fuente de datos.

DatabaseMetaData

Permite obtener información sobre la base de datos a la que te conectas.

Statement

El objeto que se usa para realizar sentencias SQL y almacena los datos que ha obtenido esta.

PreparedStatement

Sirve para almacenar una o varias sentencias SQL en él y luego utilizarlas.

CallableStatement

Sirve para llamar procedimientos almacenados en una base de datos.

ResultSet

Provee acceso a las tablas generadas mediante la realización de un Statement.

ResultSetMetaData

Permite obtener información sobre los resultados de un ResultSet.