ALMACENAMIENTO DE DATOS

1. INTRODUCCIÓN

Hasta el momento hemos estudiado el almacenamiento de datos; utilizando archivos de texto, así como tablas contendidas en bases de datos.

Es importante tener en mente los conceptos relacionados al manejo de bases de datos relacionales, y su gestión a través de lenguaje SQL.

Pero de nada serviría tener una aplicación y una base de datos; sin que se relacionaran. Recordemos que a través de los programas de nuestra aplicación, vamos a gestionar los datos almacenados en la base de datos.

Para efectuar dicha correlación de componentes, deberemos lograr enlazar nuestra aplicación con la base de datos; a través del correspondiente driver proporcionado por el fabricante del gestor de datos.

Lo interesante, es que podemos llevar instrucciones SQL a nuestras clases java e insertarlas para su correspondiente procesamiento. Estas clases Java importan librerías que nos permiten la conexión, pero además debemos agregar un String con el cual vamos a colocar todas las directrices necesarias para llegar hasta la base de datos. Dentro de estas directrices se encuentra el nombre de la base de datos, el usuario, su clave, el puerto por defecto de nuestro gestor (para el caso de nuestra herramienta postgress sql, será el puerto 5232) y la dirección del servidor.

Es necesario comprender que cada gestor de base de datos, tiene asignado un puerto por defecto, con el objetivo de evitar colisiones con otros gestores.

2. OBJETIVO DE LA UNIDAD

General:

- ✓ Implementar soluciones informáticas reales, utilizando correctamente y éticamente estructuras de almacenamiento de datos.

Específicos:

- ✓ Describir los diversos modelos JDBC para la conectividad con Bases de Datos.
- ✓ Implementar clases en lenguaje Java, que se conecten con los diversos gestores de bases de datos y le envíen instrucciones SQL.

3. ACCESANDO BASE DE DATOS RELACIONALES

En diversas aplicaciones, para lograr la conectividad entre ella y el gestor de base de datos, se utiliza un ODBC; el cual tiene las siguientes características:

- ✓ Transforma la sintaxis SQL, en una sintaxis que comprenda el gestor de bases de datos.
- ✓ De esa manera se puede cargar en la aplicación, el driver correspondiente del gestor de bases de datos, y tener la comunicación con este.
- ✓ Gracias a los drivers podemos conectarnos a infinidad de gestores de bases de datos, tales como: Postrgress SQL, Oracle, DB2, Informix, etc. Cada gestor de base de datos, tiene su correspondiente driver.

Como ya mencionamos ODBC nos permite conectar la aplicación con una base de datos, pero no tiene portabilidad como tal. Por lo que Java ideo un grupo de programas que me permiten conectar ambos componentes, pero atendiendo la portabilidad que todas las aplicaciones Java tienen.

Esta solución Java, toma el nombre de JDBC con las siguientes características:

- ✓ API de Java, que permite la conexión desde la aplicación Java hasta la base de datos, para poder gestionar la información que está dentro de ella.
- ✓ Este driver convierte el lenguaje de alto nivel, en un lenguaje que entiende la base de datos; y de esa forma gestionar sus datos.

Por lo tanto, podemos decir, que las tres acciones básicas del JDBC, son:

- ✓ Conectarse a una base de datos local o remota.
- ✓ Enviar sentencias SQL a la base de datos.
- ✓ Gestionar la información de dicha base de datos.

¿Qué hacer para que nuestra aplicación Java, se comunique con una base de datos?

- ✓ Utilizar la API de JDBC.
- ✓ Utilizar un driver JDBC, ya que este implementa la interfaz para conectarse a una base de datos.
- ✓ Deberemos agregar a nuestra aplicación, el driver correspondiente al gestor de bases de datos que estamos usando.

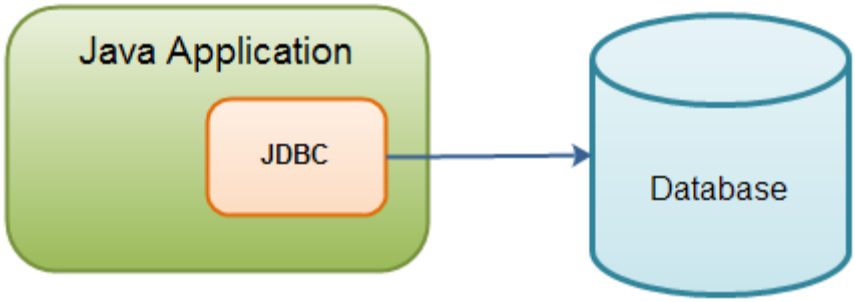
Además de los puntos anteriores, para la conectividad JDBC debemos tomar en cuenta, lo siguiente:

- ✓ JDBC, es una serie de clases y métodos que permiten a cualquier programa Java, conectarse de forma homogénea a cualquier base de datos.
- ✓ El acceso a la base de datos es a través de drivers, que son los que implementan la funcionabilidad de JDBC.
- ✓ La necesidad de JDBC para programas Java, es que este mantiene la capacidad de portabilidad; algo de lo que carece ODBC.
- ✓ ODBC debe instalarse manualmente en cada computadora; por otra parte, los drivers JDBC son portables y seguros al ser escritos directamente para Java.
- ✓ Las sentencias SQL, pasan a través de JDBC para poder gestionar la información de una base de datos.
- ✓ JDBC, permite desde una transacción simple actualizar múltiples registros de una base de datos; así como acceder a múltiples servidores de bases de datos.

- ✓ A través del connection pooling, JDBC permite ejecutar múltiples sentencias SQL, con una sola conexión.

- ✓ JDBC, permite acceder a bases de datos incompatible que corran en diferente tecnología.

- ✓ JDBC, soporta dos modelos de accesos a las bases de datos; como lo son el modelo de dos y tres capas.



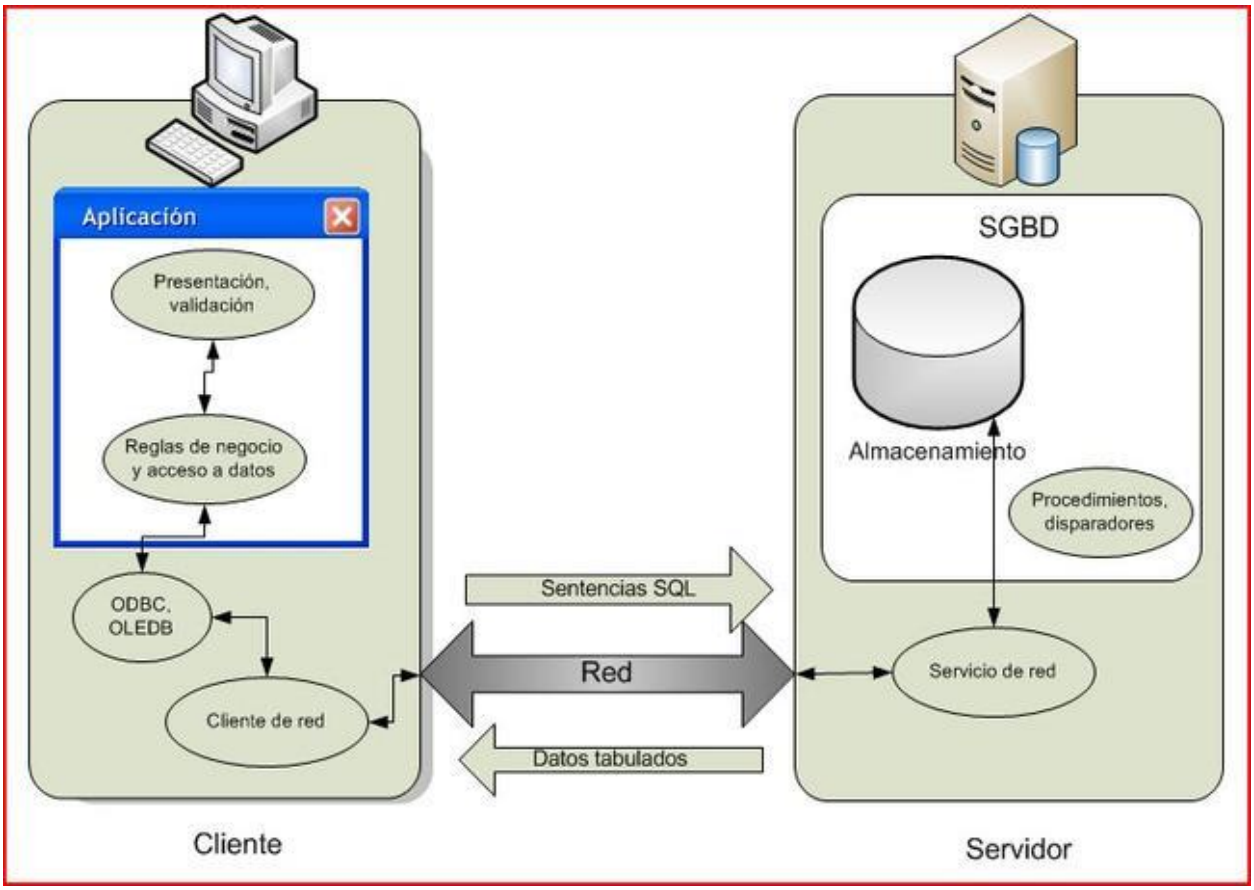
3.1 MODELO DE DOS CAPAS

Este modelo, prácticamente es una aplicación Java conectándose de forma directa a una base de datos; por lo que:

- ✓ El driver JDBC, deberá residir en el sistema local para poder conectarse a la base de datos.

- ✓ La base de datos puede estar ubicada físicamente en cualquier lugar, y se accede a ella mediante red.

- ✓ Se base en una conexión Cliente – Servidor; en la cual el programa Java envía directamente las sentencias SQL a la base de datos para la gestión de la información.



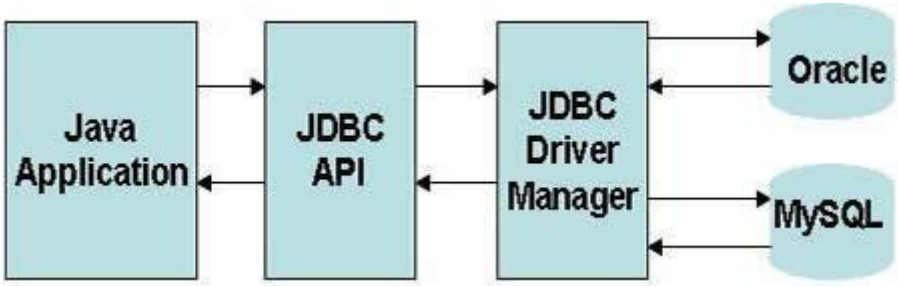
3.2 MODELO DE TRES CAPAS

Es ideal para ambientes web. Sus características son:

- ✓ Las instrucciones se envían a una capa media, entre el cliente y el servidor. Dicha capa intermedia envía las instrucciones SQL a la base de datos, y recoge los resultados de esta para exponerlos al cliente.

- ✓ El cliente no tiene contacto directo con la base de datos.

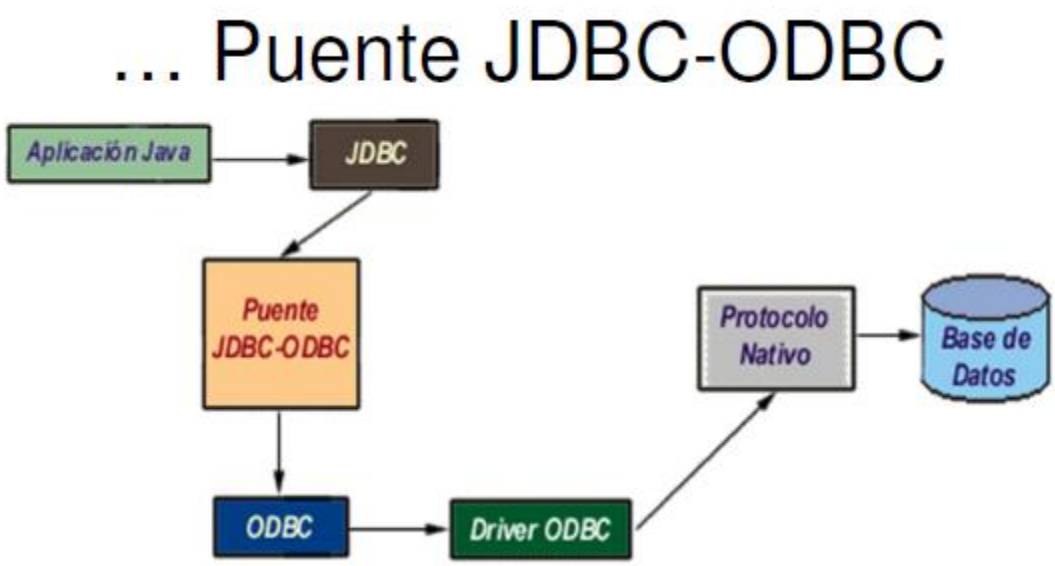
- ✓ El driver JDBC solamente reside en la capa media, que a su vez tiene todo el control con la comunicación a la base de datos.



Consideraciones:

En algunas ocasiones por temas monetarios, de infraestructura, entre otros; se decide no entrar directamente a una configuración con JDBC, sino que efectuar un puente JDBC – ODBC para poder realizar la comunicación entre aplicación y la base de datos. Importante en este caso, tener en cuenta que el puente deberá ser una solución momentánea; ya que para obtener todos los beneficios del JDBC, tendremos que migrar totalmente hacia él.

A continuación, un ejemplo de estos puentes:



Características:

- ✓ Aprovecha las configuraciones existentes; y en base a esto establece el puente JDBC-ODBC.
- ✓ Este driver hace la conversión de llamadas JDBC hacia ODBC.

Desventaja:

- ✓ Esta configuración agrega dos capas más, ya que el ODBC convierte sus llamadas, a instrucciones nativas de los gestores de bases de datos. Lo anterior puede generar algún grado de lentitud en la gestión de los datos.

3.3 CÓDIGO CON JDBC Y SQL.

- ✓ Declaración de un objeto del tipo conexión a la base de datos: Connection conexion;

- ✓ Instrucciones para cargar el Driver JDBC para SqlServer, utilizando el método forName() de la clase Class. Por ejemplo:

```
try{
   Class.forName("com.microsoft.sqlserver. jdbcSQLServerDriver").newInstance());
}
catch(Exception e){
   informacion.setText("No se pudo cargar el Driver JDBC");
}
```

Para MySQLel parámetro sería: “com.mysql.jdbc.Driver”.

Lo anterior debido a que cada gestor de bases de datos, tiene su correspondiente driver.

- ✓ La conexión al SGBD se realiza solicitándolo a la clase DriverManager. Por ejemplo:

```
StringconnectionUrl="jdbc:sqlserver://localhost:1433;" + "databaseName=pubs;user=sa;
password=adminadmin;";

conexion=DriverManager.getConnection(connectionUrl);
```

En esta sentencia podemos identificar los siguientes elementos:

- El driver del gestor sqlserver.
- La base de datos está residiendo localmente; por lo tanto, es un localhost.

- El puerto por defecto de sqlserver es el 1433. Si fuera el de Postgress sería 5432, y si trabajáramos con Mysql fuera el 3306.

- Podemos visualizar que se coloca el nombre de la base de datos, el usuario y el password.

A continuación, realizaremos un ejemplo con conexión a PostgressSQL:

✓ conexion=DriverManager.getConnection("jdbc:postgresql://localhost:5432/prn315", "prn315", "prn315");
Creamos un objeto del tipo Connection, y lo asociamos al String de conexión de PostgresSql.

✓ Statement sentencia=conexion.createStatement();
Luego creamos un objeto del tipo Statement, con el cual se van a ejecutar todas las sentencias SQL de la conexión creada previamente.

Para la ejecución de esas sentencias existen dos métodos:

- execute / executeUpdate(String sentencia)
  Ejecuta sentencias SQL, que no devuelven resultados. Por ejemplo:
  Para borrar una tabla.
  sentencia.executeUpdate(“drop table if exists datospersonales”);

Para crear una tabla.

```
sentencia.executeUpdate("create table datospersonales (“+"nombre char(20),“+"apellidos varchar(25),“+"fecha_nac date not null,“+"telefono char(10),“+"salario float )") ;
```

Para crear una llave primaria.

```
sentencia.executeUpdate("alter table datospersonales " +"add constraint pk_datosperson "+"primary key using btree(nombre, apellidos)" ) ;
```

Para insertar un registro.

```
String sql_add="INSERT INTO DATOSPERSONALES VALUES
(' " + nombre.getText().toUpperCase() + "', '" +
apellidos.getText().toUpperCase() +"', '"+ fech_nac.getText()+ "', '"
+ telefono.getText() + "', " + salario.getText()+ ") " ;

sentencia.executeUpdate(sql_add);
```

- executeQuery(String sentencia)
  Ejecuta sentencias SQL, que devuelven resultados en un objeto del tipo ResulSet.

Para seleccionar datos.

```
String sql_str = "SELECT * FROM DATOSPERSONALES" ;
ResultSet resultado=sentencia.executeQuery(sql_str);
```

Para finalizar mencionaremos, los aspectos más importantes a tomar en cuenta; cuando desarrollemos una aplicación que tenga conexión con una base de datos:

- ✓ Determinar el gestor de bases de datos que se utilizará. Generalmente esto depende de la Organización en que estemos, y del grado de transaccionabilidad que necesitemos.

- ✓ Obtener el driver del gestor de bases de datos a utilizar.

- ✓ Determinar nuestros string de conexión, en base al driver, el puerto por defecto, el nombre de la base de datos, el usuario y password.

- ✓ Agregar en la librería de nuestro proyecto, el driver correspondiente al gestor de bases de datos, para que puedan comunicarse y realizar la gestión de los datos.