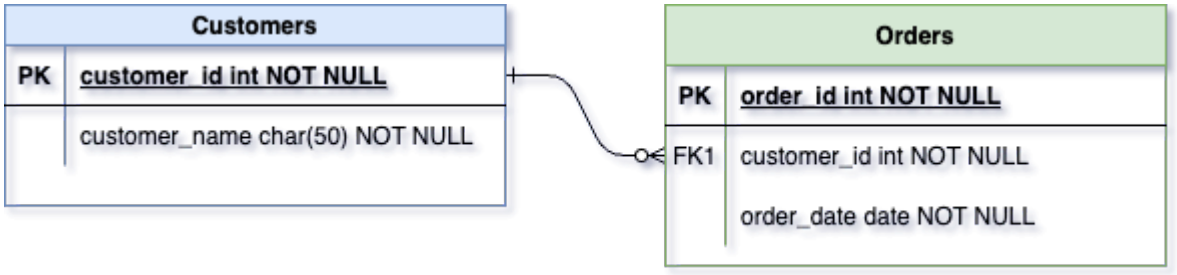
ResultSet es una de las clases más usadas cuando trabajamos con Java, específicamente cuando hacemos consultas hacía la base de datos, el gran reto con un ResultSet es poder trabajar con los resultados de la bd en una estructura más común.

Por ejemplo, supongamos que tenemos una aplicación sencilla que trabaja con una arquitectura de capas, estamos en la capa DAO y necesitamos enviar datos hacía la capa del servicio, pero enviar un ResultSet no es para nada práctico, por ende debemos enviar los datos convertidos.

En algunos casos yo opto por implementar una solución muy sencilla que es convertir el ResultSet a una Lista de Mapas es decir convertirlo a un List<Map<String, Object>> y esto lo hago obteniendo el nombre de la columna y asignándolo como la clave del mapa y el valor correspondiente lo asigno como valor para ese objeto en particular.

Por ejemplo supongamos que tenemos una estructura en BD como la siguiente:



Y ahora supongamos que tenemos un código Java que consulta a esas tablas y obtiene el ResultSet:

```
public static List<Customer> getCustomers() throws SQLException {
        List<Customer> customers = new ArrayList<>();
        String query = "SELECT customer_id, customer_name FROM
Customers";

        try (//.. Código de conexión a la BD
ResultSet resultSet = statement.executeQuery(query)) {

            // aquí vamos a convertir el ResultSet a una lista de
mapas
        }
```

```
        return customers;
    }
```

Ese código anterior es solo para tener un contexto de prueba lo que nos interesa es el siguiente código:

Primero valor a crear una clase de utilidad para garantizar podamos reutilizar este código en otras clases:

```
package com.example;

import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


public final class ResultSetUtils {

        public static List<Map<String, Object>>
resultSetToArrayList(ResultSet rs) throws SQLException {
                ResultSetMetaData md = rs.getMetaData();
                int columns = md.getColumnCount();
                List<Map<String, Object>> list = new
ArrayList<Map<String, Object>>(50);
                while (rs.next()) {
                        Map<String, Object> row = new HashMap<String,
Object>(columns);
                        for (int i = 1; i <= columns; ++i) {
                                row.put(md.getColumnName(i),
rs.getObject(i));
                        }
                        list.add(row);
                }

                return list;
```

```
        }
}
```

Explicación del código

```
ResultSetMetaData md = rs.getMetaData();
```

El método getMetaData() ese método nos permite extraer información relevante del ResultSet.

```
int columns = md.getColumnCount();
```

Ahora obtenemos cuantas columnas tiene ese resultSet, en este ejemplo son dos columnas.

```
List<Map<String, Object>> list = new ArrayList<Map<String,
Object>>(50);
```

Inicializamos una lista con un valor estimado de 50, esto no significa que vayamos a limitar la lista, solo es relevante para la inicialización y además es más optimo.

```
while (rs.next()) {
Map<String, Object> row = new HashMap<String, Object>(columns);
}
```

Después viene la iteración, creamos un mapa que por defecto va a tener el número de columnas que estamos iterando, por ejemplo si son dos columnas entonces el mapa se va a iniciar por defecto con dos registros.

```
for (int i = 1; i <= columns; ++i) {
    row.put(md.getColumnName(i), rs.getObject(i));
}
```

Dentro de la iteración sobre las filas, se itera sobre cada columna:

- md.getColumnName(i) obtiene el nombre de la columna i.
- rs.getObject(i) obtiene el valor de la columna i en la fila actual.
- row.put(md.getColumnName(i), rs.getObject(i)) añade una entrada al mapa row donde la clave es el nombre de la columna y el valor es el dato correspondiente.

```
list.add(row);
```

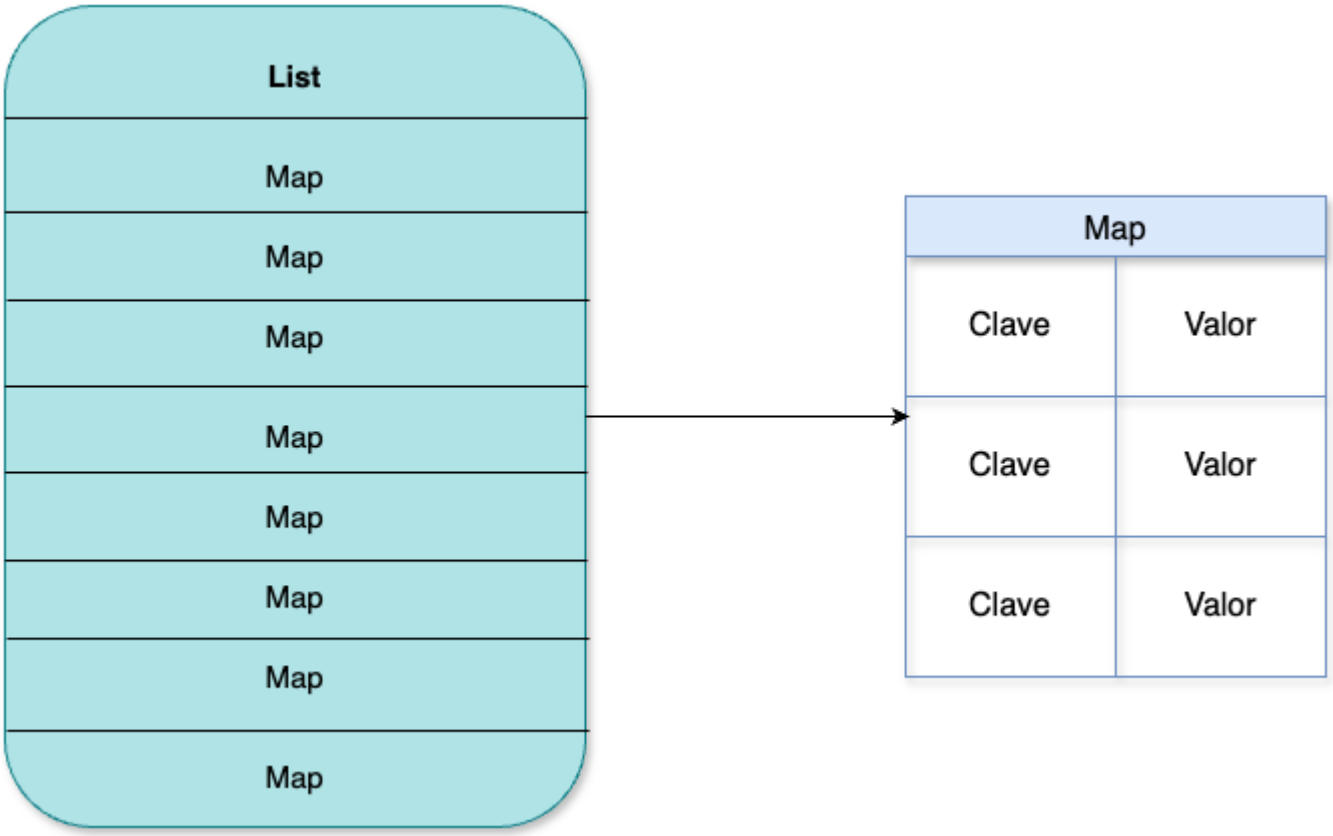
Por último añadimos el mapa construido a la Lista de mapas que creamos más arriba.

Nuestro código final sería algo así:

public static List getCustomers() throws SQLException {
List customers = new ArrayList<>();
String query = “SELECT customer_id, customer_name FROM Customers”;

```
    try (//.. Código de conexión a la BD


public static List<Customer> getCustomers() throws SQLException {
        List<Customer> customers = new ArrayList<>();
        String query = "SELECT customer_id, customer_name FROM
Customers";

        try (//.. Código de conexión a la BD
ResultSet resultSet = statement.executeQuery(query)) {

            customers = resultSetToArrayList(resultSet);
        }

        return customers;
    }
```

Representación gráfica ResultSet a List



Nuestra lista puede tener una cantidad indefinida de registros y cada uno de esos registros contiene un mapa, además dentro de cada mapa se encuentran más registros con clave y valor.

¿Cuando utilizar este código?

Recomiendo el uso del código de este ejemplo en casos donde desconozcas la estructura de la consulta, si no tienes la facilidad de convertir la estructura a una lista de un tipo de objeto determinado la solución que te brindé puede ser ideal, aunque tiene un costo en el performance no es un costo muy alto en la mayoría de los casos.