



# Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

## **Unidad de Aprendizaje:** Bases de Datos Distribuidas

**Profesor**  
De La Cruz Sosa Carlos

**Grupo**  
3TM3

**Equipo**  
6

**Alumnos**  
Rivera Ortiz Raúl Alejandro  
Solano Castrejón Eric

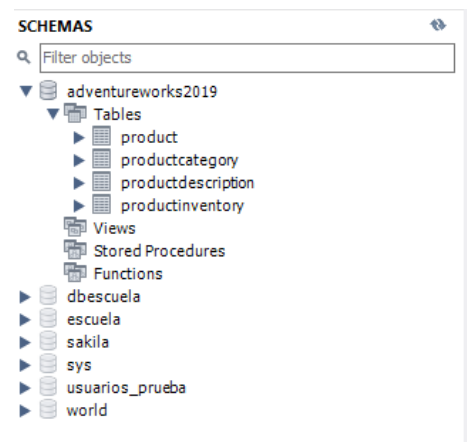
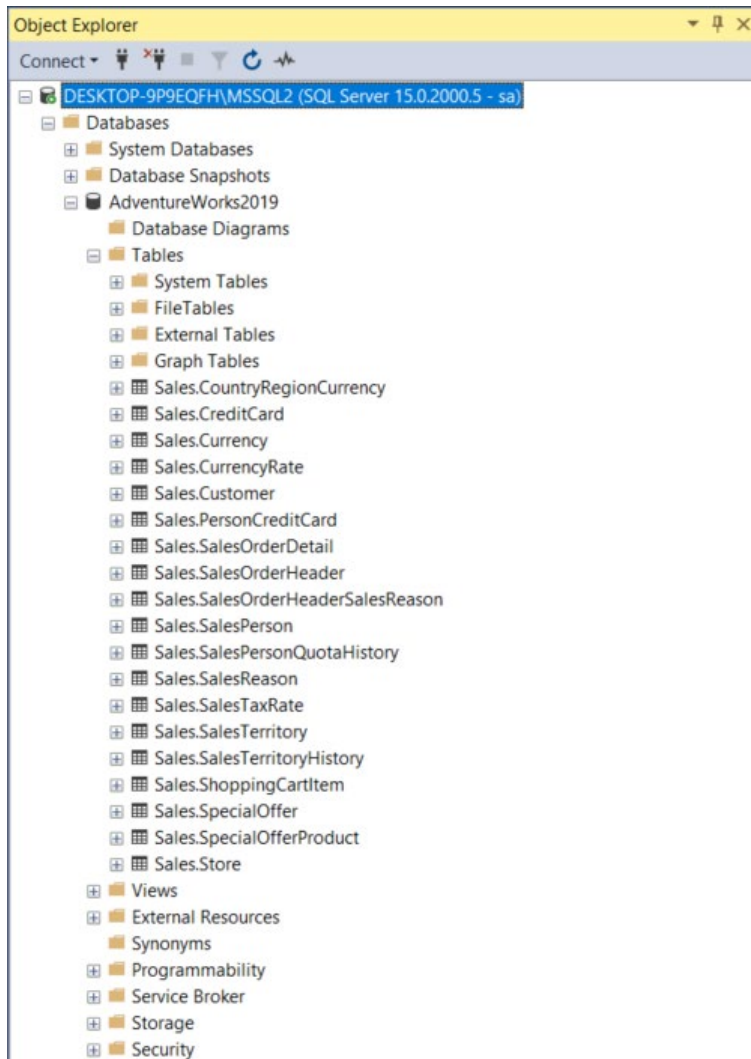
**Actividad**  
Aplicación que implementa DAO para gestionar dos  
bases de datos

## Índice

Configuración de servidores vinculados .....	3
Implementación del patrón DAO.....	8
Implementación de consultas .....	11
Implementación consultas en el DAO.....	11
Implementación INSERT SOH y SOD.....	13

## CONFIGURACIÓN DE SERVIDORES VINCULADOS

Para la realización de esta aplicación, se hace uso de dos bases de datos, una en SQL Server y otra en MySQL, la base de datos que se utiliza es AdventureWorks2019, en la instancia MySQL se alojan los datos del esquema Production, en la instancia de SQL Server se alojan los datos del esquema Sales

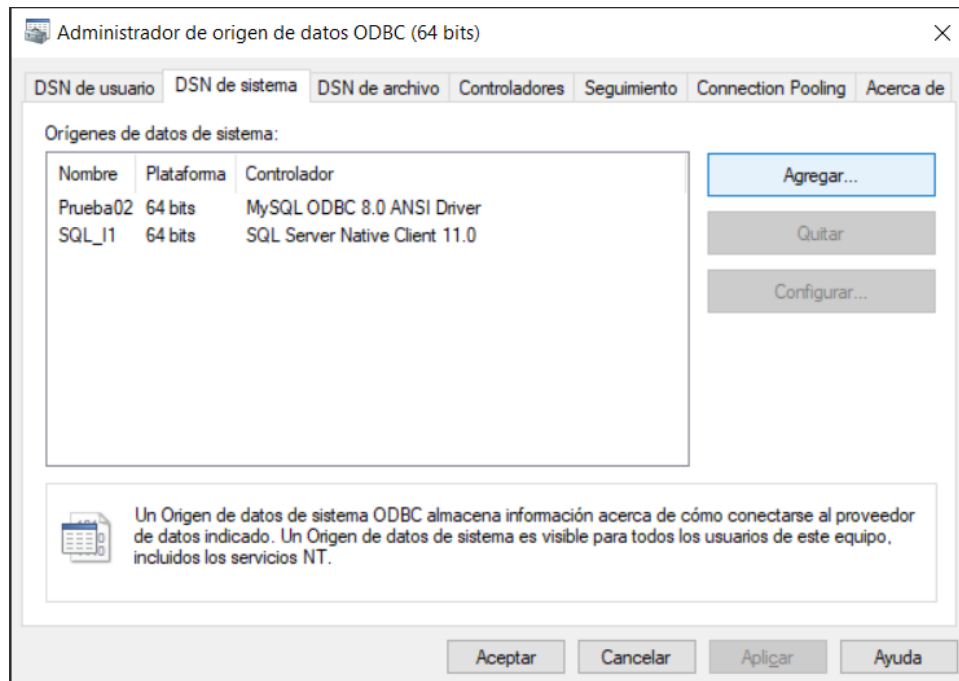


Instancia MySQL con el esquema Production

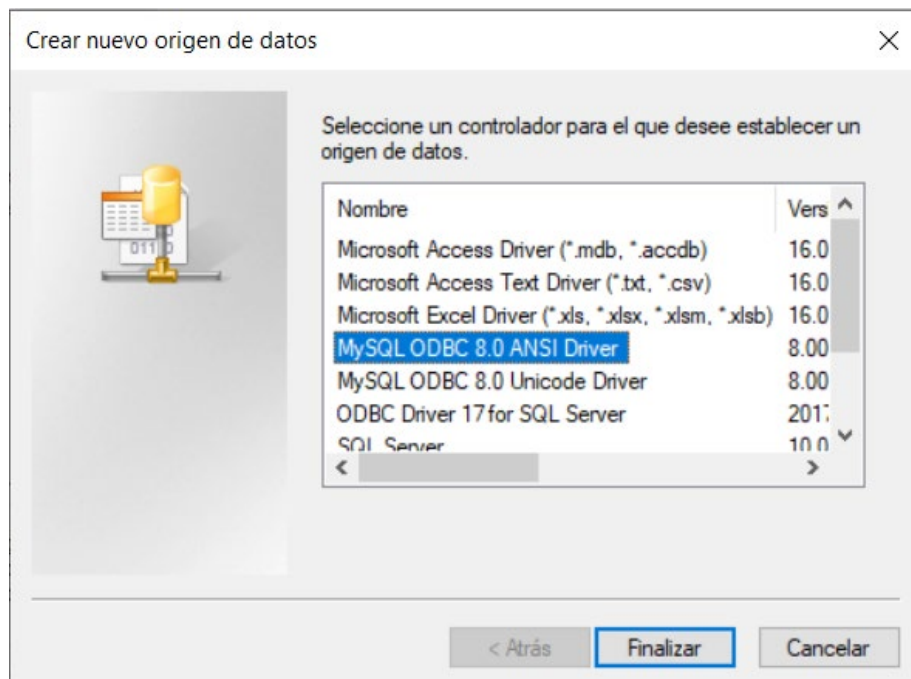
Instancia SQL Server con el esquema SALES

Ahora, para poder consultar los datos que se encuentran en MySQL desde SQL Server, tenemos que configurar un servidor vinculado, para esto tenemos que hacer los siguientes pasos.

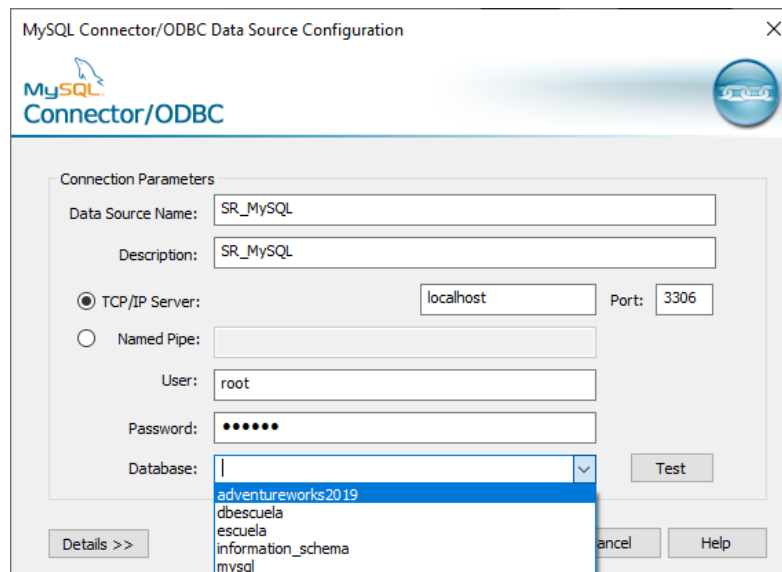
- Agregar un nuevo DSN de sistema en el administrador de origen de datos ODBC



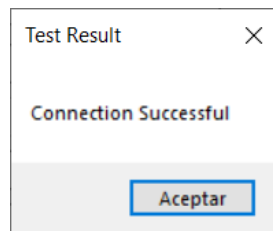
- Como nos vamos a conectar a MySQL, Seleccionamos MySQL ODBC 8.0 ---



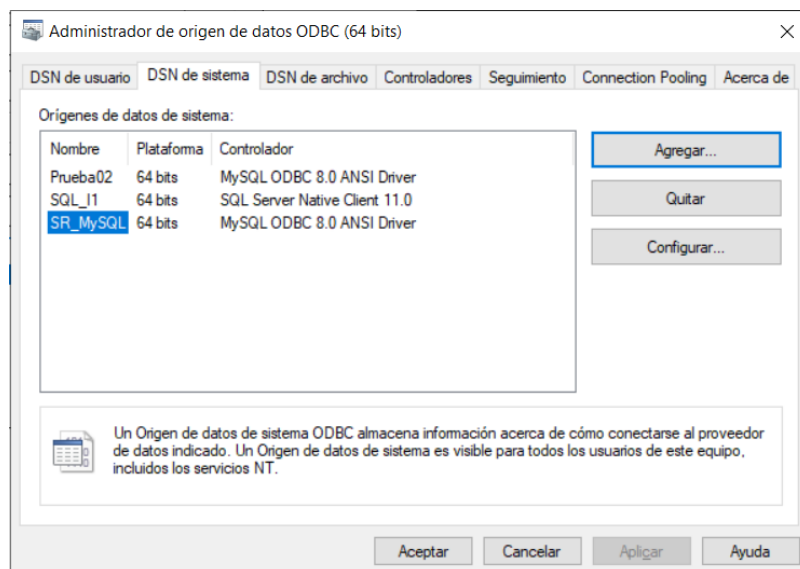
- En esta ventana ponemos el nombre y los datos para la conexión a MySQL, también seleccionamos la base de datos en donde se encuentra el esquema production



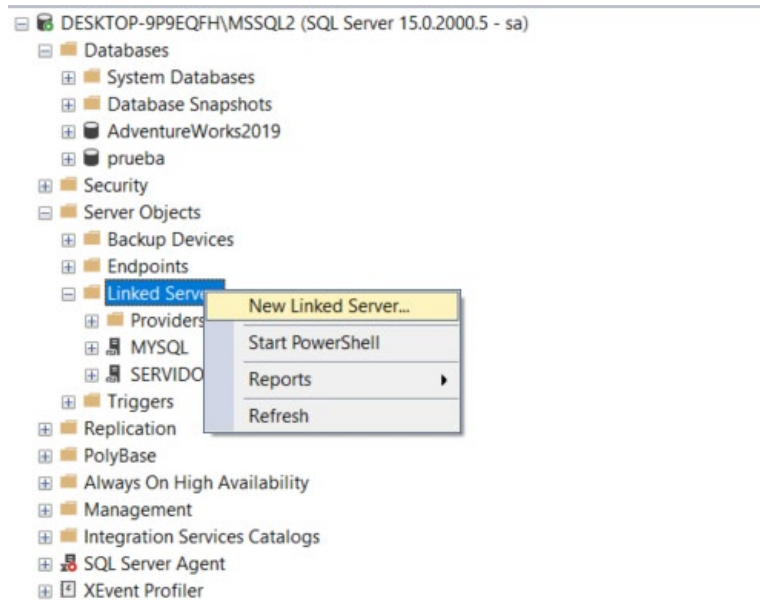
- Probamos que la conexión sea exitosa



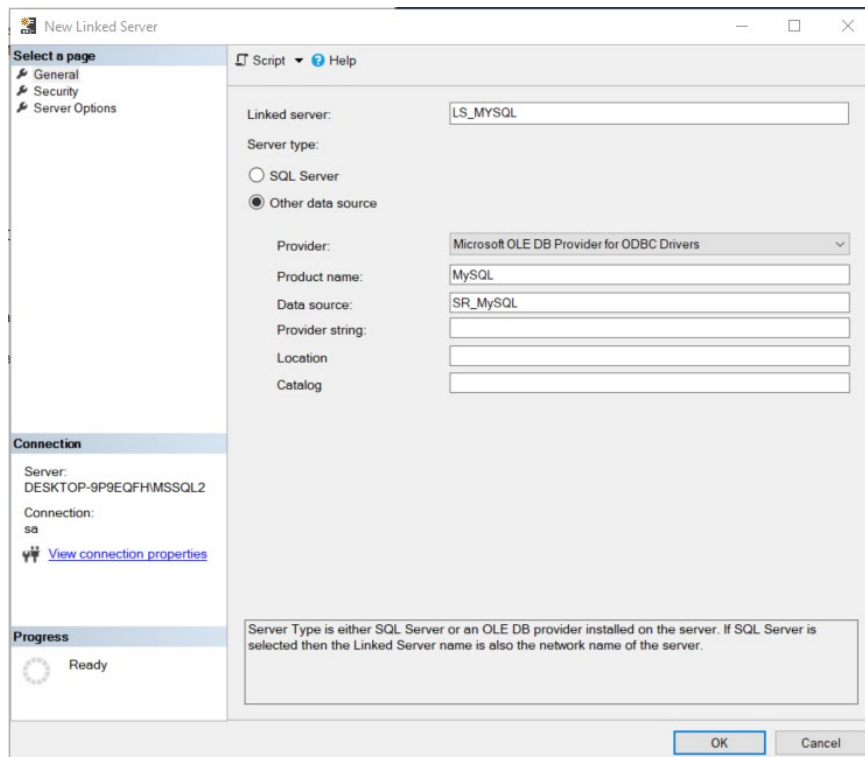
- Ya tenemos el nuevo DSN de sistema agregado



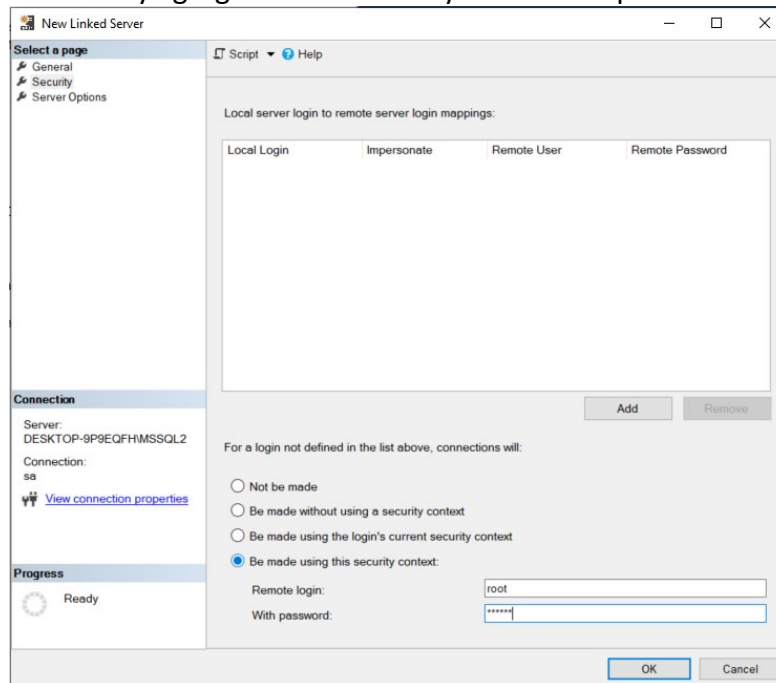
Ahora tenemos que crear el servidor vinculado en SQL Server



- Agregamos el nombre del servidor remoto
- Seleccionamos el proveedor para ODBC Drivers
- En el campo Data Source agregamos el nombre del DSN que creamos previamente



- En el apartado Security agregamos el usuario y contraseña para acceder a MySQL



Ahora tenemos creado el servidor vinculado y podemos realizar consultas a MySQL desde SQL Server, para probar que funciona correctamente este servidor vinculado, hacemos una consulta de prueba accediendo a los datos de la tabla *Product*.

Para realizar la consulta hacemos uso de OPENQUERY(), el cual recibe como primer parámetro el servidor vinculado y como segundo parámetro la consulta sql que deseemos obtener.

SQLQuery4.sql - DES...L2.master (sa (59))\*

```

1 SELECT *
2 FROM OPENQUERY(LS_MYSQL, 'SELECT * FROM PRODUCT')

```

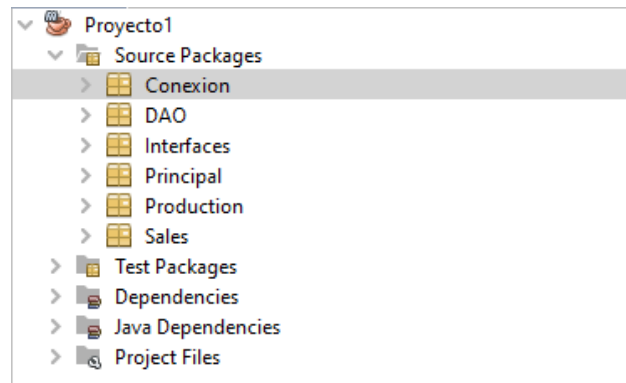
119 %

Results Messages

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice	Size
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0.0000	0.0000	NULL
2	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0.0000	0.0000	NULL
3	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0.0000	0.0000	NULL
4	4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600	0.0000	0.0000	NULL
5	316	Blade	BL-2036	1	0	NULL	800	600	0.0000	0.0000	NULL
6	317	LL Crankarm	CA-5965	0	0	Black	500	375	0.0000	0.0000	NULL
7	318	ML Crankarm	CA-6738	0	0	Black	500	375	0.0000	0.0000	NULL
8	319	HL Crankarm	CA-7457	0	0	Black	500	375	0.0000	0.0000	NULL
9	320	Chainring Bolts	CB-2903	0	0	Silver	1000	750	0.0000	0.0000	NULL
10	321	Chainring Nut	CN-6137	0	0	Silver	1000	750	0.0000	0.0000	NULL
11	322	Chainring	CR-7833	0	0	Black	1000	750	0.0000	0.0000	NULL
12	323	Crown Race	CR-9981	0	0	NULL	1000	750	0.0000	0.0000	NULL
13	324	Chain Stays	CS-2812	1	0	NULL	1000	750	0.0000	0.0000	NULL
14	325	Decal 1	DC-8732	0	0	NULL	1000	750	0.0000	0.0000	NULL
15	326	Decal 2	DC-9824	0	0	NULL	1000	750	0.0000	0.0000	NULL

## IMPLEMENTACIÓN DEL PATRÓN DAO

Una vez teniendo los esquemas correspondientes en cada instancia y el servidor vinculado, usamos el patrón DAO para acceder a la información de dichas instancias, esto lo haremos desde JAVA



Antes que nada, tenemos que establecer conexión con las instancias donde se encuentran las tablas que estaremos accediendo.

Tenemos que crear la cadena de conexión en la cual se especifica el driver que estamos usando, el servidor al cual nos tratamos de conectar, así como el puerto, el nombre de la base de datos, el usuario y contraseña

```
public class ConexionSQLserver {

    protected Connection conexion;

    //Variables para acceder a la base de datos
    private final String usuario = "sa";
    // private final String contraseña = "qwerty19";
    private final String contraseña = "990699";
    private final String bd = "AdventureWorks2019";
    private final String ip = "DESKTOP-9P9EQFH\\MSSQL2";
    private final String puerto = "1434";//Puerto de la segunda instancia

    private final String JDBC_DRIVER = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    private final String BD_URL = "jdbc:sqlserver://" + ip + ":" + puerto + ";" + "databaseName=" + bd;

    public void conectar() throws Exception {
        try {
            conexion = DriverManager.getConnection(BD_URL, usuario, contraseña);
            //Class.forName(JDBC_DRIVER);

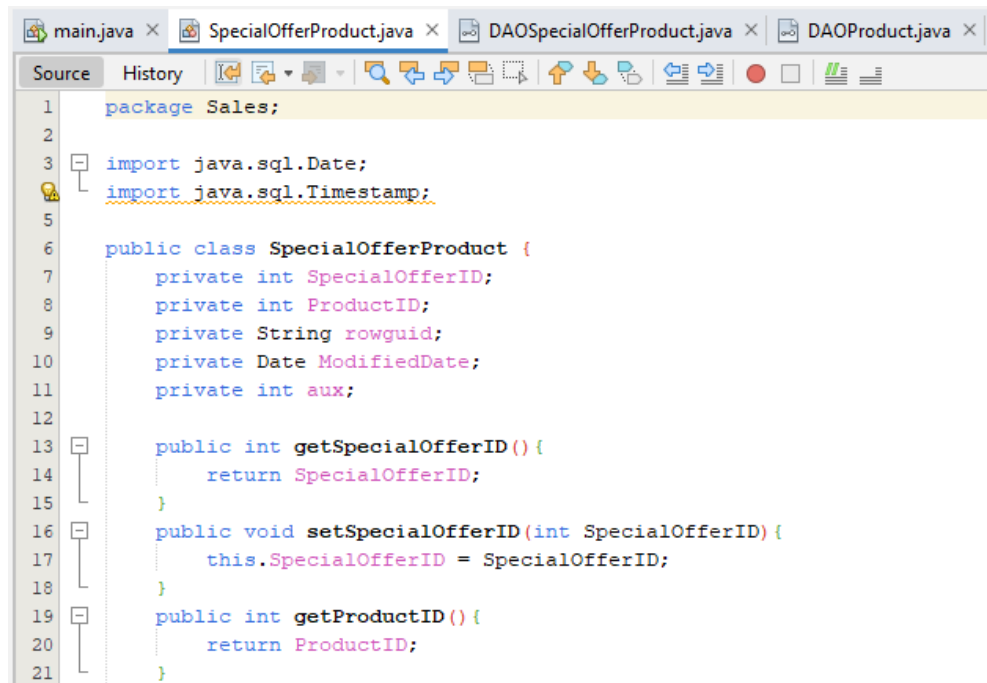
            // System.out.println("CONECTADO CORRECTAMENTE SQLSERVER");
            //System.out.println("Conexion con SQLserver exitosa");

        } catch (SQLException e) {
            System.out.println("ERROR " + e);
        }
    }
}
```



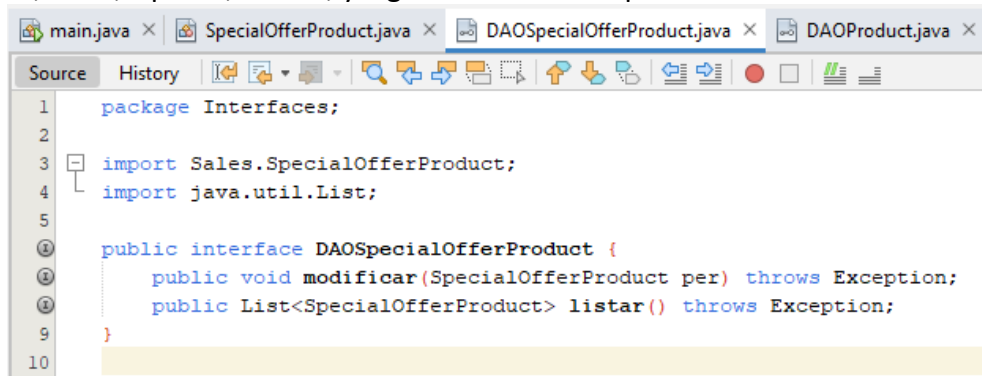
Para implementar el patron DAO, tenemos que:

- Crear las clases de cada tabla, en esta clase estarán los atributos de las tablas y sus respectivos Get() y Set(), como se muestra en la siguiente imagen.



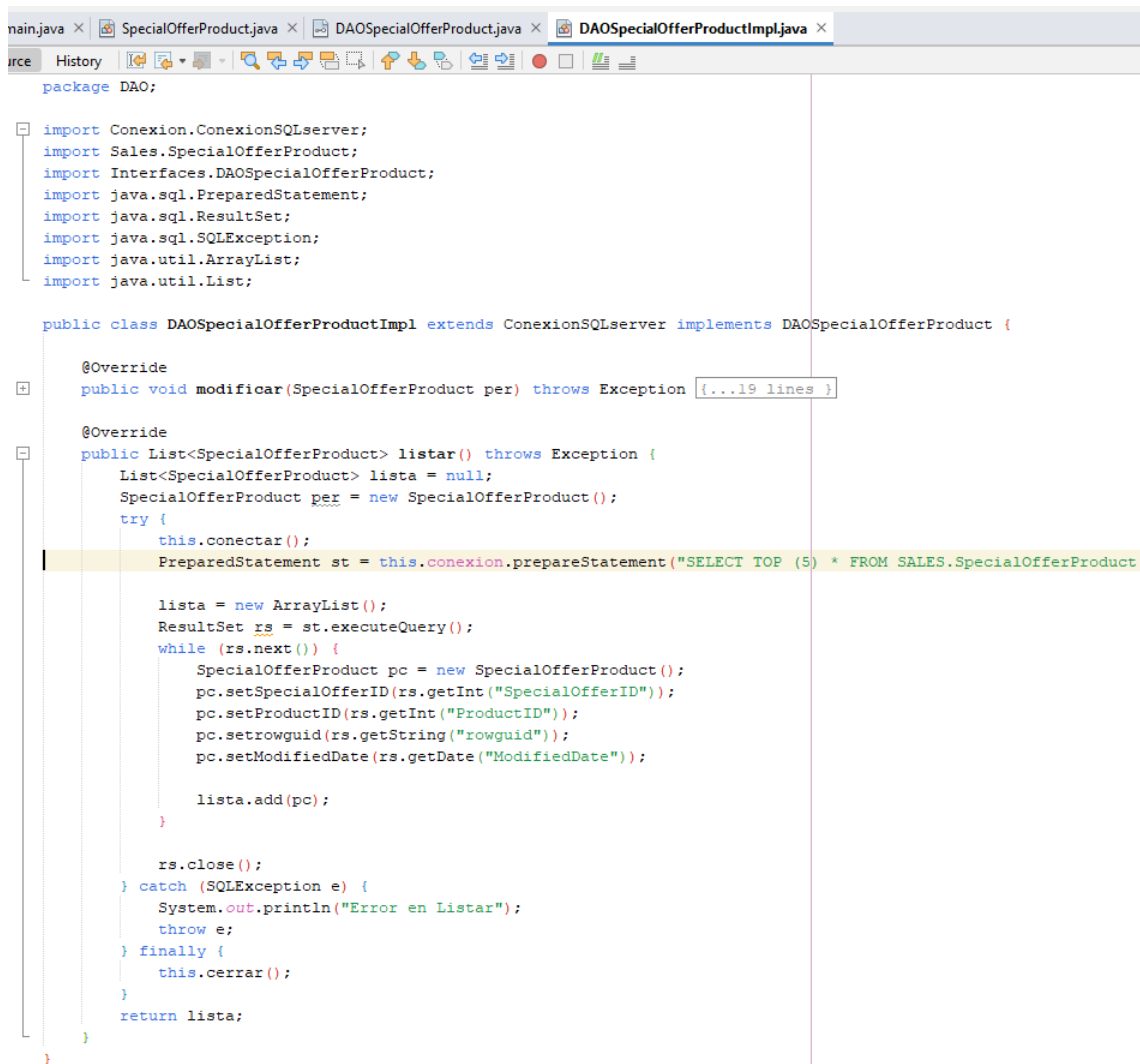
```
1 package Sales;
2
3 import java.sql.Date;
4 import java.sql.Timestamp;
5
6 public class SpecialOfferProduct {
7     private int SpecialOfferID;
8     private int ProductID;
9     private String rowguid;
10    private Date ModifiedDate;
11    private int aux;
12
13    public int getSpecialOfferID() {
14        return SpecialOfferID;
15    }
16    public void setSpecialOfferID(int SpecialOfferID) {
17        this.SpecialOfferID = SpecialOfferID;
18    }
19    public int getProductID() {
20        return ProductID;
21    }
22 }
```

- Creamos las interfaces en donde especificaremos los métodos que vamos a usar tales como Create, Read, Update, Delete, y algún otro método que consideremos útil.



```
1 package Interfaces;
2
3 import Sales.SpecialOfferProduct;
4 import java.util.List;
5
6 public interface DAOSpecialOfferProduct {
7     public void modificar(SpecialOfferProduct per) throws Exception;
8     public List<SpecialOfferProduct> listar() throws Exception;
9 }
10
```

- Posteriormente creamos otra clase donde implementaremos los métodos y como estaremos haciendo consultas a las instancias MySQL y SQL Server estas deben extender la clase Conexión e implementar la interfaz correspondiente.



```
package DAO;

import Conexion.ConexionSQLServer;
import Sales.SpecialOfferProduct;
import Interfaces.DAOSpecialOfferProduct;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class DAOSpecialOfferProductImpl extends ConexionSQLServer implements DAOSpecialOfferProduct {

    @Override
    public void modificar(SpecialOfferProduct per) throws Exception { ...19 lines }

    @Override
    public List<SpecialOfferProduct> listar() throws Exception {
        List<SpecialOfferProduct> lista = null;
        SpecialOfferProduct per = new SpecialOfferProduct();
        try {
            this.conectar();
            PreparedStatement st = this.conexion.prepareStatement("SELECT TOP (5) * FROM SALES.SpecialOfferProduct");

            lista = new ArrayList();
            ResultSet rs = st.executeQuery();
            while (rs.next()) {
                SpecialOfferProduct pc = new SpecialOfferProduct();
                pc.setSpecialOfferID(rs.getInt("SpecialOfferID"));
                pc.setProductID(rs.getInt("ProductID"));
                pc.setrowguid(rs.getString("rowguid"));
                pc.setModifiedDate(rs.getDate("ModifiedDate"));

                lista.add(pc);
            }

            rs.close();
        } catch (SQLException e) {
            System.out.println("Error en Listar");
            throw e;
        } finally {
            this.cerrar();
        }
        return lista;
    }
}
```

En este punto es el que cambia mas para cada clase, ya que aquí es donde accedemos a los datos de cada tabla, por lo tanto, las consultas SQL tienen que ser específicas para obtener los datos correctos.

Una vez tenemos estas clases, podemos implementar el main() en donde podremos mostrar el resultado de las consultas como se muestra a continuación:

```
DAO SpecialOfferProduct
1. CREATE
2. READ
3. UPDATE
4. DELETE
0. SALIR
Escribe una de las opciones
2
CONECTADO CORRECTAMENTE SQLSERVER
Conexion con SQLServer exitosa
1 -- 810 -- 4EF0DF05-C5BE-4EFA-8815-14285800DDF1 -- 2022-06-05
1 -- 921 -- E7FCC6D3-DC00-4A6A-A299-505D5B48DFD1 -- 2022-06-05
1 -- 922 -- 89E4CEDF-4D8E-4FF9-92C4-3597A0407A0A -- 2022-06-05
1 -- 706 -- B3C9A4B1-2AE6-4CBA-B552-1F206C9F4C1F -- 2022-06-05
2 -- 680 -- BB30B868-D86C-4557-8DB2-4B2D0A83A0FB -- 2022-06-05
```

## IMPLEMENTACIÓN DE CONSULTAS

### Implementación consultas en el DAO

Como se mencionó anteriormente, las clases DAO\_\_Impl tienen que hacer extend a la clase Conexión, pues en esta se encuentran los métodos necesarios para acceder a la correspondiente base de datos.

```
@Override
public List<SpecialOfferProduct> listar() throws Exception {
    List<SpecialOfferProduct> lista = null;
    SpecialOfferProduct per = new SpecialOfferProduct();
    try {
        this.conectar();
        PreparedStatement st = this.conexion.prepareStatement("SELECT TOP (5) * FROM SALES.SpecialOfferProduct");

        lista = new ArrayList();
        ResultSet rs = st.executeQuery();
        while (rs.next()) {
            SpecialOfferProduct pc = new SpecialOfferProduct();
            pc.setSpecialOfferID(rs.getInt("SpecialOfferID"));
            pc.setProductID(rs.getInt("ProductID"));
            pc.setrowguid(rs.getString("rowguid"));
            pc.setModifiedDate(rs.getDate("ModifiedDate"));

            lista.add(pc);
        }

        rs.close();
    } catch (SQLException e) {
        System.out.println("Error en Listar");
        throw e;
    } finally {
        this.cerrar();
    }
    return lista;
}
```

Una vez obtenemos el objeto conexión, podemos implementar las consultas, en este caso se muestra el ejemplo de un *SELECT*:

- Usando **PreparedStatement** en donde se especifica la consulta SQL a realizar
- Para ejecutar la consulta, se utiliza **ExecuteQuery**
- Tenemos que guardar los resultados en **ResultSet**
- Recorrer **ResultSet** hasta que se llegue al final de los resultados,
  - Estos resultados los vamos guardando en el objeto correspondiente, aquí hacemos uso de los **Set()** que se crearon para cada atributo
  - Se agregan a una lista los objetos **SpecialOfferProduct** que posteriormente, en el **main()** los mostramos en consola
- Finalmente se cierra la conexión con el servidor

Este proceso es similar para las distintas consultas que se realizan, por ejemplo, para realizar un UPDATE en SpecialOfferProduct, se tiene que hacer lo siguiente

```
14      @Override
15      public void modificar(SpecialOfferProduct per) throws Exception {
16          try {
17              this.conectar();
18              PreparedStatement st = this.conexion.prepareStatement(
19                  "UPDATE SALES.SpecialOfferProduct SET SpecialOfferID = ?,rowguid=NEWID(),"
20                  + " ModifiedDate = SYSDATETIME() where SpecialOfferID = ? AND ProductID = ? ";
21              st.setInt(1, per.getSpecialOfferID()); //Nuevo SpecialOfferID
22              st.setInt(2, per.getAux()); //Antiguo SpecialOfferID
23              st.setInt(3, per.getProductID()); //
24
25              st.executeUpdate();
26              System.out.println("Modificado Correctamente");
27          } catch (Exception e) {
28              System.out.println("Error en Modificar");
29              throw e;
30          } finally {
31              this.cerrar();
32          }
33      }
34  }
```

- Abrir la conexión con el servidor
- Especificar la consulta a realizar en el **PreparedStatement**
  - Especificar las “variables” que se van a sustituir en la consulta en donde se encuentra un símbolo “?”, estas las obtenemos del objeto SpecialOfferProduct con su respectivo get() para cada atributo, estos atributos son proporcionados por el usuario
- Ejecutar la consulta
- Cerrar la conexión

Una vez tenemos todos los métodos para las clases que se utilizan para esta aplicación, podemos implementarlas en el main().

## Implementación INSERT SOH y SOD

A diferencia de las consultas anteriores, que las implementábamos en la parte de JAVA de la aplicación, esta inserción decidimos implementarla con Stored Procedures (SP) que se encuentran en la instancia de SQL Server, dichos SP hacen consultas distribuidas a la instancia de MySQL, esto con el Servidor vinculado que se mostro como configurar al inicio de este documento.

Para realizar el insert a las tablas SOH y SOD, se hace uso de 2 Stored Procedure

- sp\_insert\_OrderDetail
- sp\_insert\_OrderHeader

En este documento se explicará a grandes rasgos que es lo que hacen dichos SP, el Query realizado para la creación de estos se encuentra en el repositorio de GitHub, dicho Query se encuentra comentado para ayudar a entender la solución propuesta por el equipo.

### sp\_insert\_OrderDetail

En este SP se realizan diversas consultas distribuidas para determinar si el producto que ingresa el usuario existe y si la cantidad que solicita está disponible.

Cuando se cumplen esas condiciones

Se procede a verificar si existe algún descuento para dicho producto.

Se hace otra consulta distribuida para obtener el precio del producto

Finalmente se hace el insert correspondiente

Cuando no se cumplen las condiciones

Se hacen consultas distribuidas para determinar que condición no se cumple para retornar un valor y con dicho valor especificar al usuario cual es la condición que no se cumple

Este stored procedure tiene diversas variables de entrada que nos proporcionan la información necesaria para realizar las consultas y variables de salida que, nos proporciona el id de la inserción además de que, en caso de algún error, nos proporciona un valor correspondiente a cuál fue el error que ocurrió.

### **sp\_insert\_OrderHeader**

En este stored procedure no se realizan consultas distribuidas, pues los datos que necesitamos acceder se encuentran en la misma instancia y fueron insertados por el SP anterior.

La consulta que realiza este es más sencilla pues solo requerimos obtener el valor del total de la/ las inserciones que se realizaron en el otro SP, para esto se utiliza la variable de entrada ID que fue una de las variables de salida del SP anterior.

Al igual que el otro SP, este stored procedure tiene diversas variables que son necesarias para la inserción.

## Implementación usuario

Una vez explicados los elementos anteriores, ahora podemos implementar la aplicación.

Se realizo un menú en el cual podremos acceder las consultas anteriormente mencionadas

```
***** Selecciona una opcion *****
1. Sales
2. Product
3. Inserción SOH y SOD
0. Salir
Escribe una de las opciones
```

### Opción 1: Sales

Al seleccionar la opción 1 se despliega otro menú en donde podemos escoger a cuál tabla del esquema sales queremos acceder

```
1
1. Sales.Customer
2. Sales.OrderDetail
3. Sales.OrderHeader
4. Sales.SpecialOfferProduct
Escribe una de las opciones
4
DAO SpecialOfferProduct
1. CREATE
2. READ
3. UPDATE
4. DELETE
0. SALIR
Escribe una de las opciones
2
CONECTADO CORRECTAMENTE SQLSERVER
Conexion con SQLserver exitosa
1 -- 810 -- 4EF0DF05-C5BE-4EFA-8815-14285800DDF1 -- 2022-06-05
1 -- 921 -- E7FCC6D3-DC00-4A6A-A299-505D5B48DFD1 -- 2022-06-05
1 -- 922 -- 89E4CEDF-4D8E-4FF9-92C4-3597A0407A0A -- 2022-06-05
1 -- 706 -- B3C9A4B1-2AE6-4CBA-B552-1F206C9F4C1F -- 2022-06-05
2 -- 680 -- BB30B868-D86C-4557-8DB2-4B2D0A83A0FB -- 2022-06-05
```

En esta imagen se muestra los datos que se obtienen de una consulta de tipo lectura a la tabla SpecialOfferProduct.

## Opción 2: Product

Al seleccionar la opción 2 se despliega otro menú en donde podemos escoger a cuál tabla del esquema Product queremos acceder.

```
2
1. Product
2. ProductCategory
3. ProductDescription
4. ProductInventory
Escribe una de las opciones
```

```
1
DAO Product
```

```
1. CREATE
2. READ
3. UPDATE
4. DELETE
0. SALIR
```

```
Escribe una de las opciones
```

```
1
```

```
Create
```

```
Insert Product
Ingrese el Nombre
Prueba Insert
Ingrese el ProductNumber
Ingrese el SafetyStockLevel
100
Ingrese el ReorderPoint
50
Ingrese el StandardCost
1250
Ingrese el ListPrice
1500
Registrado Correctamente
```

```
Registrado Correctamente
```

```
1. CREATE
2. READ
3. UPDATE
4. DELETE
0. SALIR
```

```
Escribe una de las opciones
```

```
2
```

```
Read
```

```
1000 -- Prueba -- Insert -- 1250.0
999 -- Road-750 Black, 52 -- BK-R19B-52 -- 343.6497
998 -- Road-750 Black, 48 -- BK-R19B-48 -- 343.6496
997 -- Road-750 Black, 44 -- BK-R19B-44 -- 343.6496
996 -- HL Bottom Bracket -- BB-9108 -- 53.9416
995 -- ML Bottom Bracket -- BB-8107 -- 44.9506
994 -- LL Bottom Bracket -- BB-7421 -- 23.9716
993 -- Mountain-500 Black, 52 -- BK-M18B-52 -- 294.5797
992 -- Mountain-500 Black, 48 -- BK-M18B-48 -- 294.5797
991 -- Mountain-500 Black, 44 -- BK-M18B-44 -- 294.5797
```

En esta imagen se muestra los datos que se insertaron en la tabla Product con la consulta realizada por el usuario.

En esta imagen se muestra un ejemplo de una insercion a la tabla Product, vemos que los datos son proporcionados por el usuario y al final se muestra que se realizo la insercion de manera correcta



## Opción 2: Inserción SOH y SOD

Al seleccionar la opción 3 se inicia el proceso de inserción a SOD y SOH

```
3
Proceso de insercion a SOD y SOH
Ingresa el producto:
921
Ingresa la cantidad:
10
CONECTADO CORRECTAMENTE SQLSERVER
Conexion con SQLserver exitosa
->Precio: 4.99
->OrderID= 75145
Quieres Agregar otro producto?
1-> SI
0-> NO
1
Ingresa el producto:
811
Ingresa la cantidad:
2
CONECTADO CORRECTAMENTE SQLSERVER
Conexion con SQLserver exitosa
->Precio: 44.54
->OrderID= 75145
Quieres Agregar otro producto?
1-> SI
0-> NO
1
Ingresa el producto:
15697
Ingresa la cantidad:
10000
CONECTADO CORRECTAMENTE SQLSERVER
Conexion con SQLserver exitosa
->Precio: -1
EL PRODUCTO NO EXISTE
Quieres Agregar otro producto?
1-> SI
0-> NO
0
Se registro correctamente
```

Primero se solicita al usuario ingresar el producto y la cantidad, esto será insertado en SOD

Se verifica la existencia del producto y se obtiene el OrderID que se utiliza para la inserción a SOH

Se realiza este proceso hasta que no se desee hacer más inserts a SOD

Una vez ya no se van a agregar mas registros a SOD, se procede a hacer el registro a SOH

```
Se registro correctamente

Sales Order Detail
CONECTADO CORRECTAMENTE SQLSERVER
Conexion con SQLserver exitosa
75145 -- 121352 -- 4911-403C-98 -- 10 -- 921
75145 -- 121353 -- 4911-403C-98 -- 2 -- 811

Sales Order Header
CONECTADO CORRECTAMENTE SQLSERVER
Conexion con SQLserver exitosa
75145 -- 5 -- 153.5729 -- 2022-06-06
```

En esta imagen se muestran los registros insertados de la imagen anterior.