



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 de Fragmentación de bases de datos

6 de junio de 2022

Índice

Configuración de servidores vinculados	11
Implementación del patrón DAO.....	16
Implementación de consultas	19
Implementación consultas en el DAO.....	19
Implementación INSERT SOH y SOD.....	¡Error! Marcador no definido.

FRAGMENTACION HORIZONTAL

Consultas a considerar para la fragmentación

1. La información de los clientes de almacenarse por región, considerando las regiones de acuerdo con el atributo group de SalesTerritory
2. Listar datos del empleado que atendió mas ordenes por territorio
3. Listar los datos del cliente con mas ordenes solicitadas en la región "North America"
4. Listar el producto mas solicitado en la región "Europe"
5. Listar las ofertas que tienen mas productos de la categoría "Bikes"
6. Listar los 3 productos menos solicitados en la región "Pacific"
7. Actualizar la subcategoria de los productos con ProductID del 1 al 4 a la subcategoria válida para el tipo de producto.
8. Listar los productos que no estén disponibles a la venta.
9. Listar los clientes del territorio 1 al 4 que no tengas asociado un valor en PersonID
10. Listar los clientes del territorio 1 que tengan ordenes en otro territorio.

Consultas que generan fragmentos de customer a partir de l conjunto M

PrCustomer={

P1: TerritoryID = 1

...

P10: TerritoryID=10

}

MCustomer= {

M1: $P1 \wedge P2 \wedge P3 \wedge P4 \wedge P5 \wedge P6 \wedge \neg(P7) \wedge \neg(P8) \wedge \neg(P9) \wedge \neg(P10)$

M2: $\neg(P1) \wedge \neg(P2) \wedge \neg(P3) \wedge \neg(P4) \wedge \neg(P5) \wedge \neg(P6) \wedge P7 \wedge P8 \wedge P9 \wedge \neg(P10)$

M3: $\neg(P1) \wedge \neg(P2) \wedge \neg(P3) \wedge \neg(P4) \wedge \neg(P5) \wedge \neg(P6) \wedge \neg(P7) \wedge \neg(P8) \wedge \neg(P9) \wedge P10$

M4: $\neg(P1) \wedge \neg(P2) \wedge \neg(P3) \wedge \neg(P4) \wedge \neg(P5) \wedge \neg(P6) \wedge \neg(P7) \wedge \neg(P8) \wedge \neg(P9) \wedge \neg(P10)$

}

Customer_F1 =TerritoryID1-6(Customer)

SELECT*FROM Customer where TerritoryID BETWEEN 1 and 6

Customer _F2 =TerritoryID=7,8,10(Customer)

SELECT*FROM Customer where TerritoryID BETWEEN 7 and 9

Customer _F3 =TerritoryID=9(Customer)

SELECT*FROM Customer where TerritoryID = 10

Customer _F4 =TerritoryID1-6(Customer)

SELECT*FROM Customer where TerritoryID > 10

Consultas semijoin que generan los fragmentos de SalesOrderHeader y SalesOrderDetail a partir de la fragmentación horizontal primaria de Customer

Fragmento 1

SalesOrderHeader_F1 = SalesOrderHeader Customer_F1

SalesOrderHeader (TerritoryID1-6)(Customer_F1))

```
SELECT c.* INTO Customer
FROM AdventureWorks2019.Sales.Customer c
where TerritoryID BETWEEN 1 AND 6
go
```

```
SELECT DISTINCT soh.* INTO SalesOrderHeader
FROM AdventureWorks2019.Sales.SalesOrderHeader soh
JOIN (SELECT *FROM Pacifico.dbo.Customer) c
ON soh.TerritoryID = c.TerritoryID
```

Fragmento 2

SalesOrderHeader_F2 = SalesOrderHeader Customer_F2

SalesOrderHeader (TerritoryID1-6)(Customer_F2))

```
SELECT c.* INTO Customer
FROM AdventureWorks2019.Sales.Customer c
where TerritoryID = 7 OR TerritoryID = 8 OR TerritoryID = 10
go
```

```
--Fragmento SalesOrderHeader Norteamerica
SELECT DISTINCT soh.* INTO SalesOrderHeader
FROM AdventureWorks2019.Sales.SalesOrderHeader soh
JOIN (SELECT *FROM Europa.dbo.Customer) c
ON soh.TerritoryID = c.TerritoryID
GOGO
```

Fragmento 3

SalesOrderHeader_F3 = SalesOrderHeader Customer_F3

SalesOrderHeader_F3 = SalesOrderHeader (TerritoryID10)(Customer_F3))

```
SELECT c.* INTO Customer
FROM AdventureWorks2019.Sales.Customer c
where TerritoryID= 9
go
```

```
--Fragmento SalesOrderHeader Norteamerica
SELECT DISTINCT soh.* INTO SalesOrderHeader
FROM AdventureWorks2019.Sales.SalesOrderHeader soh
JOIN (SELECT *FROM Pacifico.dbo.Customer) c
ON soh.TerritoryID = c.TerritoryID
GO
```

Fragmento 4

SalesOrderHeader_F4: SalesOrderHeader Customer_F4

```

SOH_F4 = SalesOrderHeader (TerritoryID>10)(Customer_F4))
SELECT *
FROM Sales.SalesOrderHeader
Where CustomerID IN(Select CustomerID FROM Sales.Customer
Where TerritoryID > 10)

```

Fragmentación de SalesOrderDetail a partir de los fragmentos de SalesOrderHeader

Fragmento 1

```

SalesOrderDetail _F1 = SalesOrderDetail SalesOrderHeader_F1
SalesOrderDetail _F1 = (SalesOrderID=SalesOrderID (SalesOrderHeader_F1))
SELECT DISTINCT sod.* INTO SalesOrderDetail
FROM AdventureWorks2019.Sales.SalesOrderDetail sod
JOIN (SELECT * FROM Norteamerica.dbo.SalesOrderHeader) st
ON sod.SalesOrderID = st.SalesOrderID
GO

```

Fragmento 2

```

SalesOrderDetail _F2= SalesOrderDetail SalesOrderHeader_F2
SalesOrderDetail _F2 = (SalesOrderID=SalesOrderID (SalesOrderHeader_F2))
SELECT DISTINCT sod.* INTO SalesOrderDetail
FROM AdventureWorks2019.Sales.SalesOrderDetail sod
JOIN (SELECT * FROM Europa.dbo.SalesOrderHeader) st
ON sod.SalesOrderID = st.SalesOrderID

```

Fragmento 3

```

SalesOrderDetail _F3 = SalesOrderDetail SalesOrderHeader_F3
SalesOrderDetail _F3 = (SalesOrderID=SalesOrderID (SalesOrderHeader_F3))
SELECT DISTINCT sod.* INTO SalesOrderDetail
FROM AdventureWorks2019.Sales.SalesOrderDetail sod
JOIN (SELECT * FROM Pacifico.dbo.SalesOrderHeader) st
ON sod.SalesOrderID = st.SalesOrderID

```

Fragmento 4

```

SalesOrderDetail _F4 = SalesOrderDetail SalesOrderHeader_F4
SalesOrderDetail _F4 = (SalesOrderID=SalesOrderID (SalesOrderHeader_F4))

```

Fragmentación del esquema Production a partir de los enunciados

```

PrProductCategory={
    P: ProductCategoryID = 1 ---Categoría Bikes
}
PrSubcategory={

```

```

P1: ProductSubcategoryID = 1
P2: ProductSubcategoryID = 2
P3: ProductSubcategoryID = 3
}
PrProduct={
    P1: ProductSubcategoryID = 1
    P2: ProductSubcategoryID = 2
    P3: ProductSubcategoryID = 3
    P4: ProductID=1
    P5: ProductID=2
    P6: ProductID=3
    P7: ProductID=4
    P8: SellEndDate IS NULL
}
MProduct{
    M1: P1^P2^P3^¬(P4)^¬(P5)^¬(P6)^¬(P7)^¬(P8)    --Productos de categoría bikes
    M2: ¬(P1) ^¬(P2) ^¬(P3) ^P4 ^P5 ^P6^P7^¬(P8)    --Actualización ProductID 4 al 7
    M3: ¬(P1) ^¬(P2) ^¬(P3) ^¬(P4) ^¬(P5) ^¬(P6) ^¬(P7) ^P8 -- Productos no a la venta
}

```

Consultas que generan fragmentos de esquema production

SpecialOfferProduct_F = SpecialOfferProduct Product

SpecialOfferProduct_F = (ProductSubCategoryID Between 1 and 3 (Product))

SELECT *FROM Sales.SpecialOfferProduct as sop

where sop.ProductID IN(

Select p.ProductID FROM Product as p

where p.ProductSubcategoryID BETWEEN 1 AND 3)

Fragmentación propuesta

Analizando posible fragmentación por esquema Production

De la obtención de los predicados a partir de los enunciados podemos darnos cuenta de que en el caso de fragmentar la base de datos a partir del esquema production podemos darnos cuenta desde la obtención del primer predicado que no cumple el ser completo y mínimo ya que solo hace referencia a una categoría de productos (Bikes) y deja fuera todas

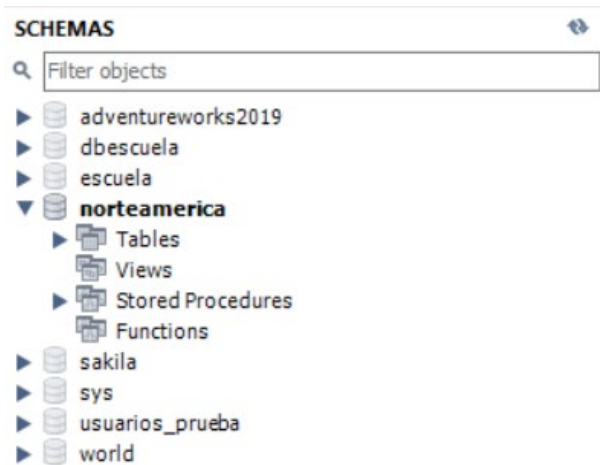
las demás, además de que solamente una consulta hace referencia a ese atributo, lo que también nos hace concluir que no es relevante por lo que no tendría sentido fragmentar por este criterio.

ASIGNACION DE FRAGMENTOS

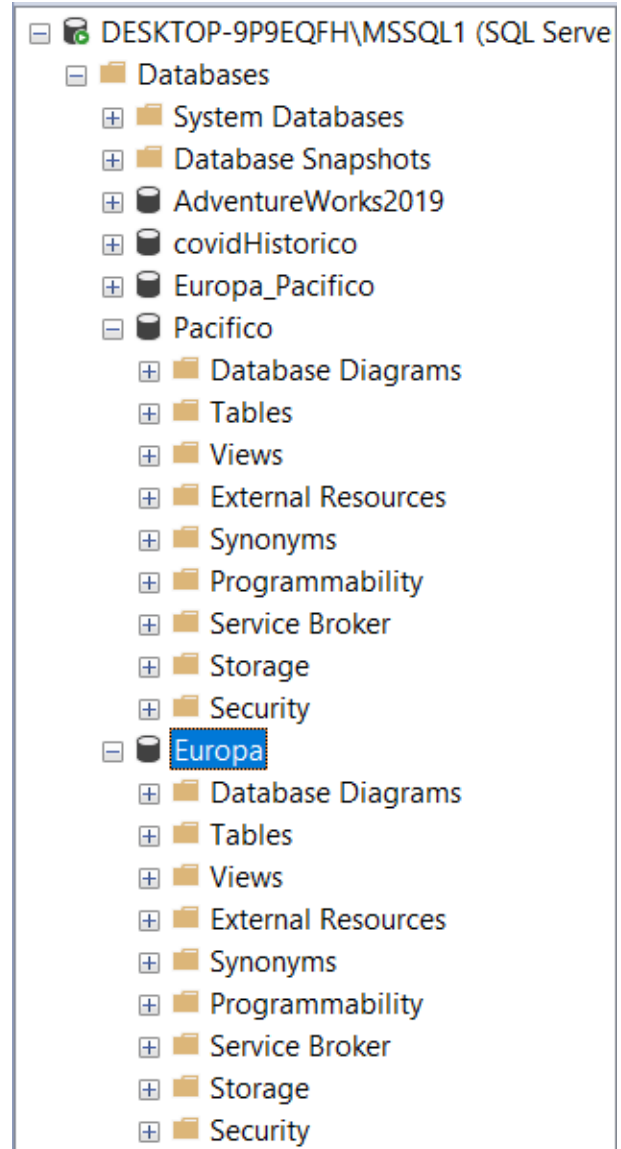
Como se menciona en el apartado anterior, se crean 4 fragmentos dependiendo del Territorio, ahora los fragmentos creados los vamos a alojar en MySQL y SQLServer.

El fragmento de NorteaAmerica se alojan en MySQL

Los fragmentos Europa y Pacifico se alojan en SQLServer



El 4 fragmento es para nuevas regiones, como las consultas que se deben implementar son de lectura y aún no se cuenta con ningún dato que no entre en este fragmento, solamente se realizó la asignación con los demás fragmentos



Si bien ya tenemos fragmentos hechos, falta también determinar qué tablas no fragmentadas son necesarias para la realización de las consultas.

Las tablas no fragmentadas que son necesarias son:

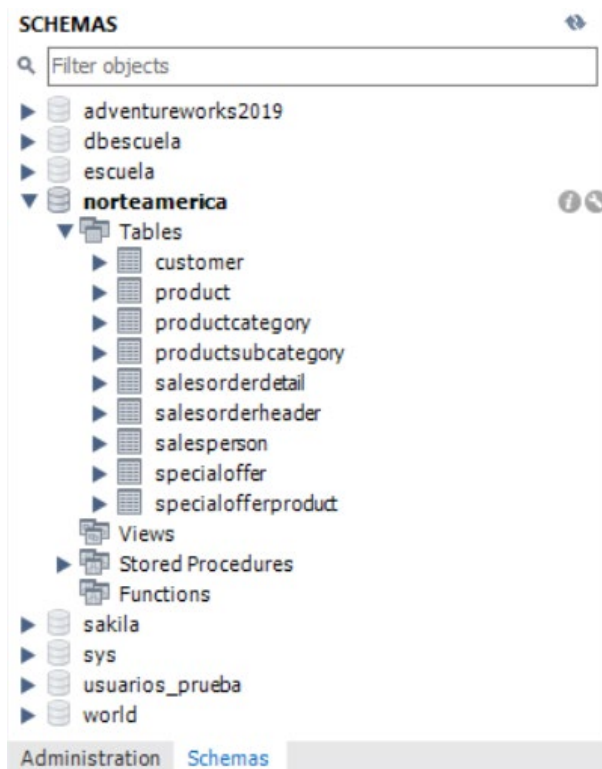
- SpecialOffer
- SpecialOfferProduct
- SalesPerson
- Product
- ProductCategory
- ProductSubcategory

Ahora surgía una cuestión y es en donde se deben almacenar estas tablas.

Encontramos dos opciones

- Replicar las tablas en cada servidor
- Almacenarlas en un solo servidor y acceder por medio de servidor vinculado

Optamos por replicar las tablas en cada servidor, pues las consultas son de tipo lectura y tener los datos que se necesitan para estas en el mismo servidor ayudará a que el acceso a la información sea más rápido.



En la instancia MySQL decidimos insertar las tablas no fragmentadas en la misma base NorteAmérica, pues no tenemos más fragmentos como lo es para el caso de SQLServer

DESKTOP-9P9EQFH\MSSQL1 (SQL Serve

Databases

System Databases

Database Snapshots

AdventureWorks2019

covidHistorico

Europa_Pacifico

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Product

dbo.ProductCategory

dbo.ProductSubcategory

dbo.SalesPerson

dbo.SpecialOffer

dbo.SpecialOfferProduct

Views

External Resources

Synonyms

Programmability

Service Broker

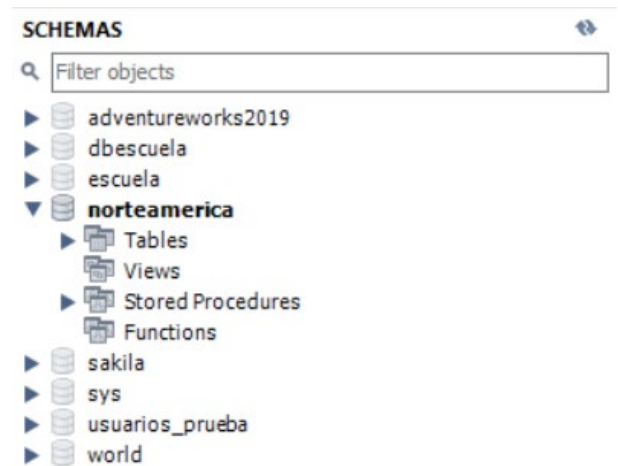
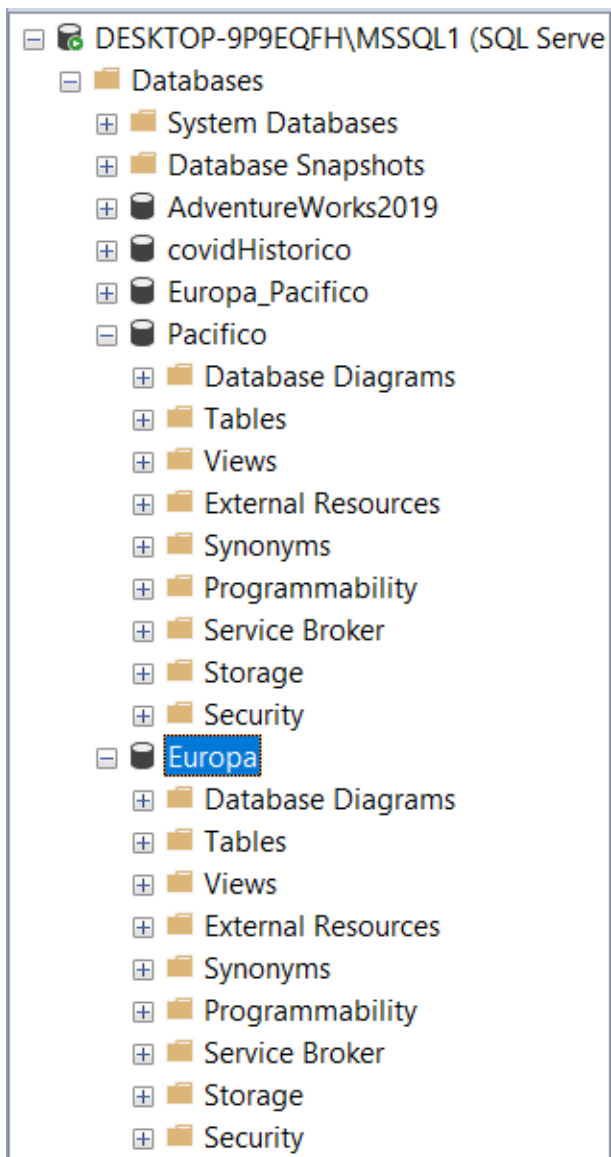
Storage

Security

Para la instancia de SQLServer decidimos insertar estas tablas en una nueva base de datos, por lo tanto cuando se haga alguna consulta que necesite información de las tablas no fragmentadas, se tendrá que hacer uso de la base de datos Europa_Pacifico para acceder a los datos.

CONFIGURACIÓN DE SERVIDORES VINCULADOS

Diversas de las consultas que se deben realizar en este proyecto, están descritas de tal modo en el que no es necesario implementar consultas distribuidas para darle solución, además, como decidimos replicar las tablas no fragmentadas a los distintos servidores, la información la tenemos en las tablas, pero hay algunas consultas que requieren datos de los fragmentos, por lo tanto se tiene que configurar un servidor vinculado para resolver dichas consultas.

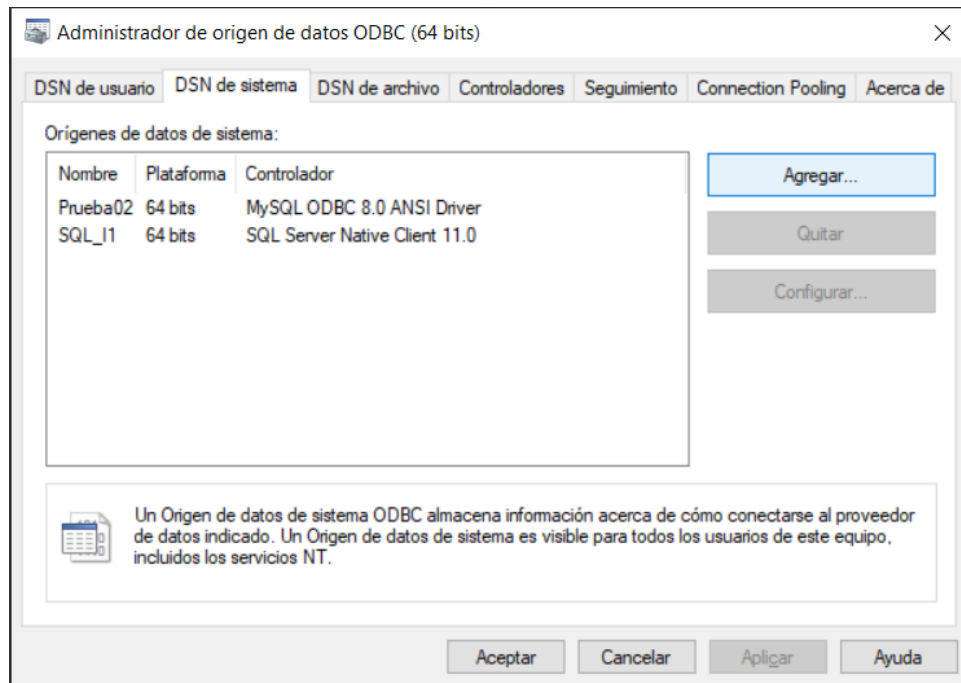


Instancia MySQL el fragmento
NorteAmerica

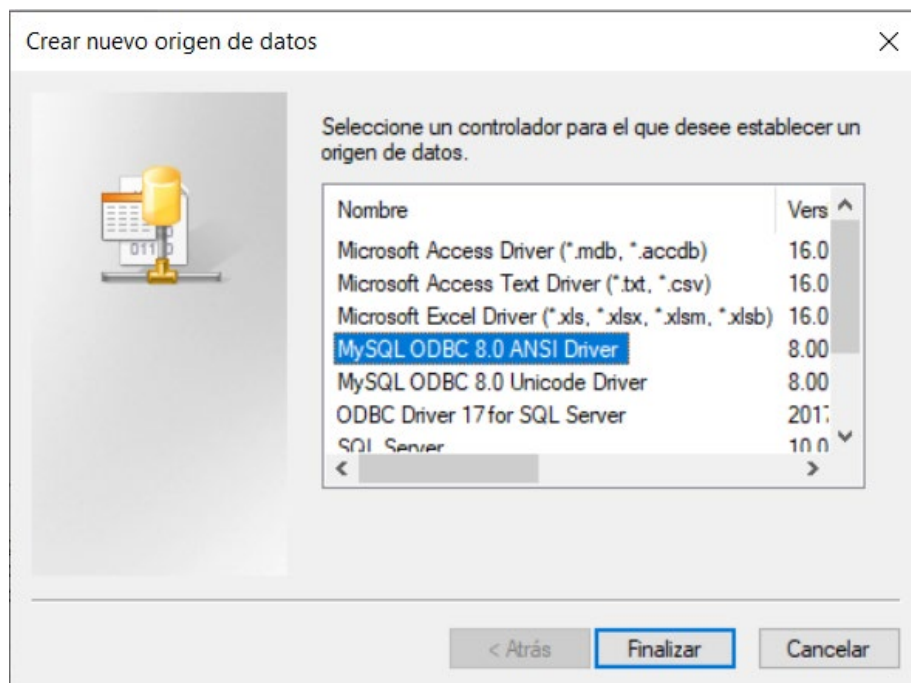
Instancia SQL Server con los fragmentos Europa y Pacifico

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.

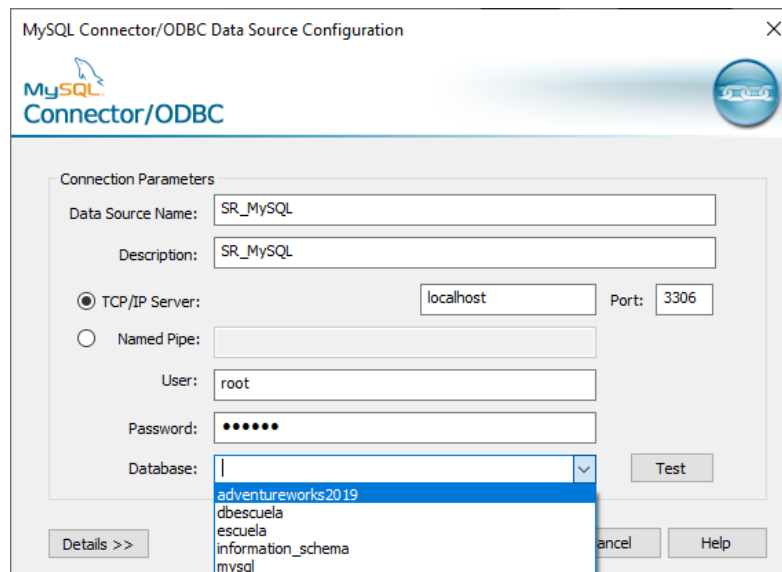
- Agregar un nuevo DSN de sistema en el administrador de origen de datos ODBC



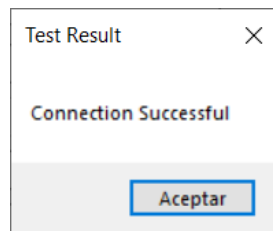
- Como nos vamos a conectar a MySQL, Seleccionamos MySQL ODBC 8.0 ---



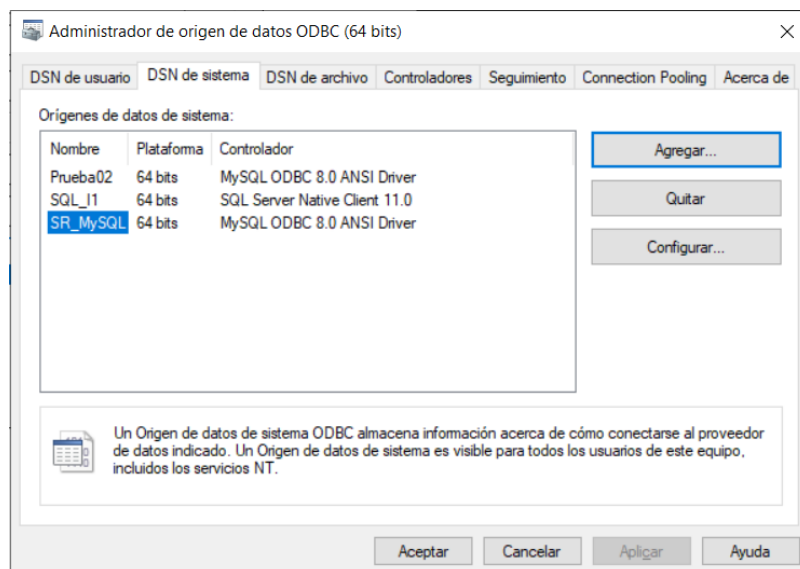
- En esta ventana ponemos el nombre y los datos para la conexión a MySQL, también seleccionamos la base de datos a acceder



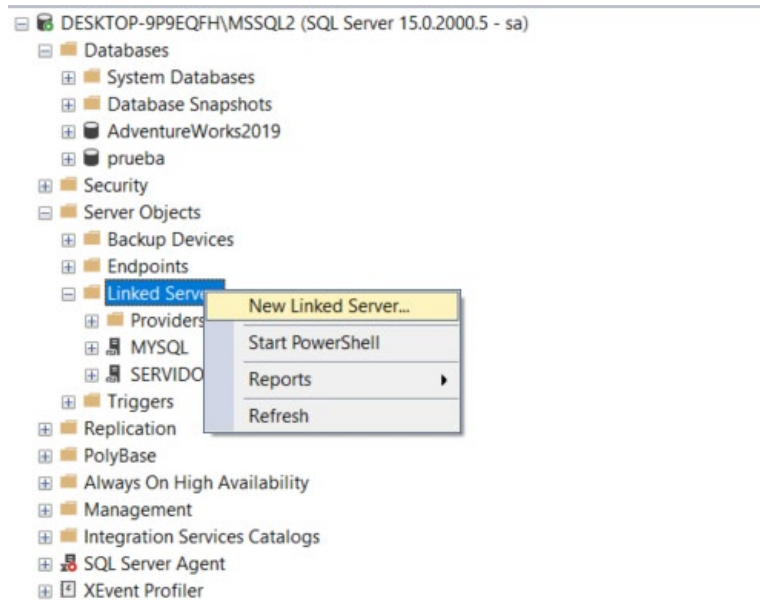
- Probamos que la conexión sea exitosa



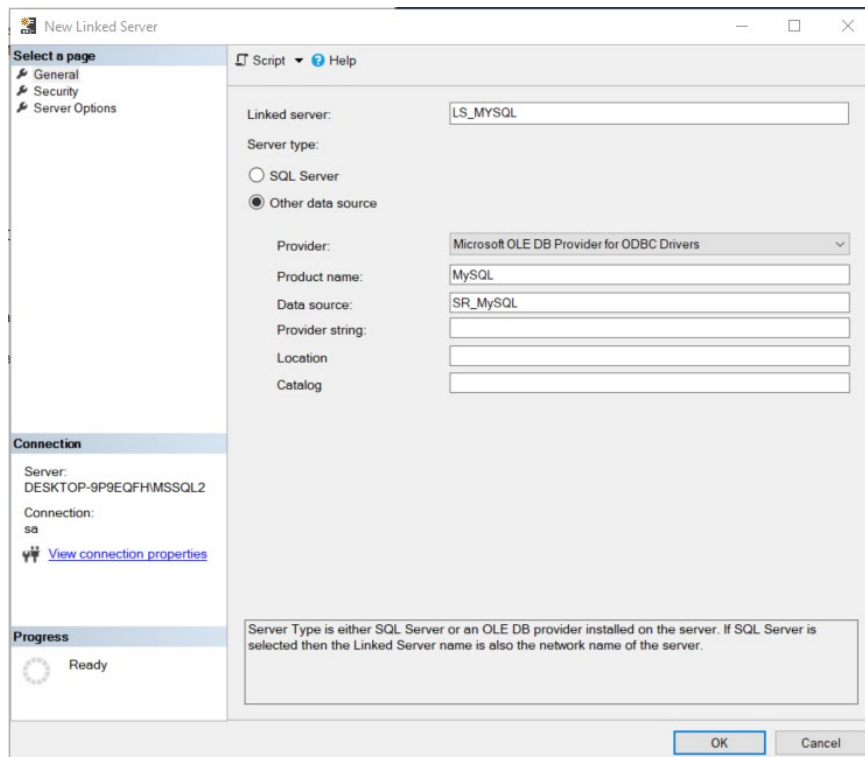
- Ya tenemos el nuevo DSN de sistema agregado



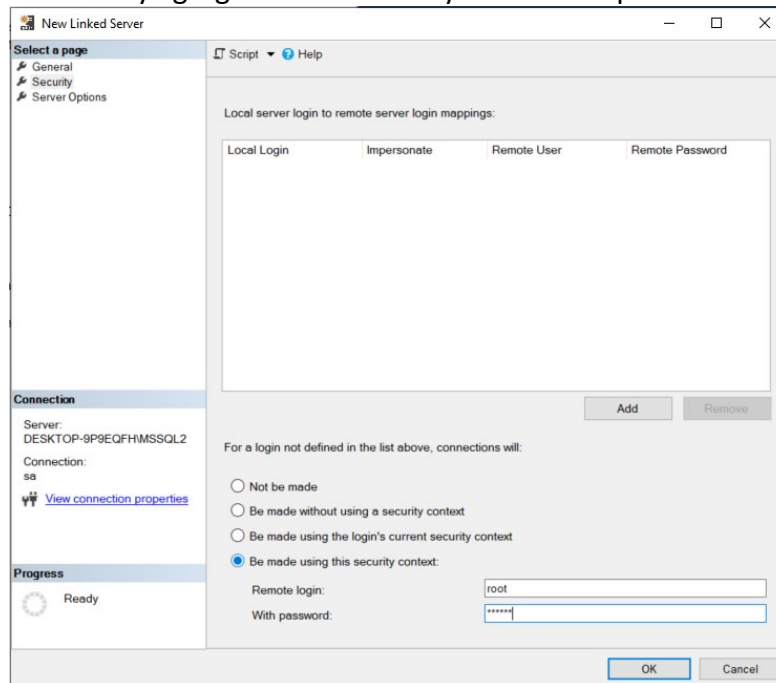
Ahora tenemos que crear el servidor vinculado en SQL Server



- 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')

```

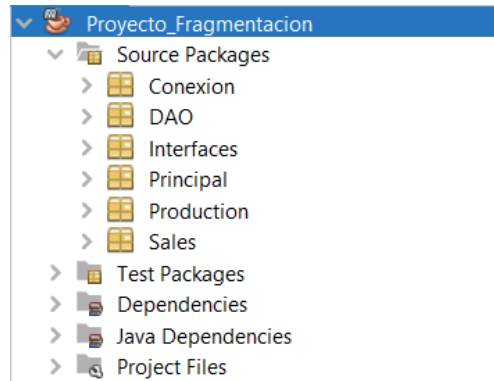
119 %

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 DE LAS CONSULTAS

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 ConexionMySQL {

    protected Connection conexion;

    //Variables para acceder a la base de datos
    private final String usuario = "root";
    // private final String contraseña = "";
    private final String contraseña = "990699";
    private final String bd = "norteamerica";
    private final String ip = "localhost";
    private final String puerto = "3306";

    private final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver"; //Antiguo com.mysql.jdbc.Driver
    private final String BD_URL = "jdbc:mysql://" + ip + ":" + puerto + "/" + bd;

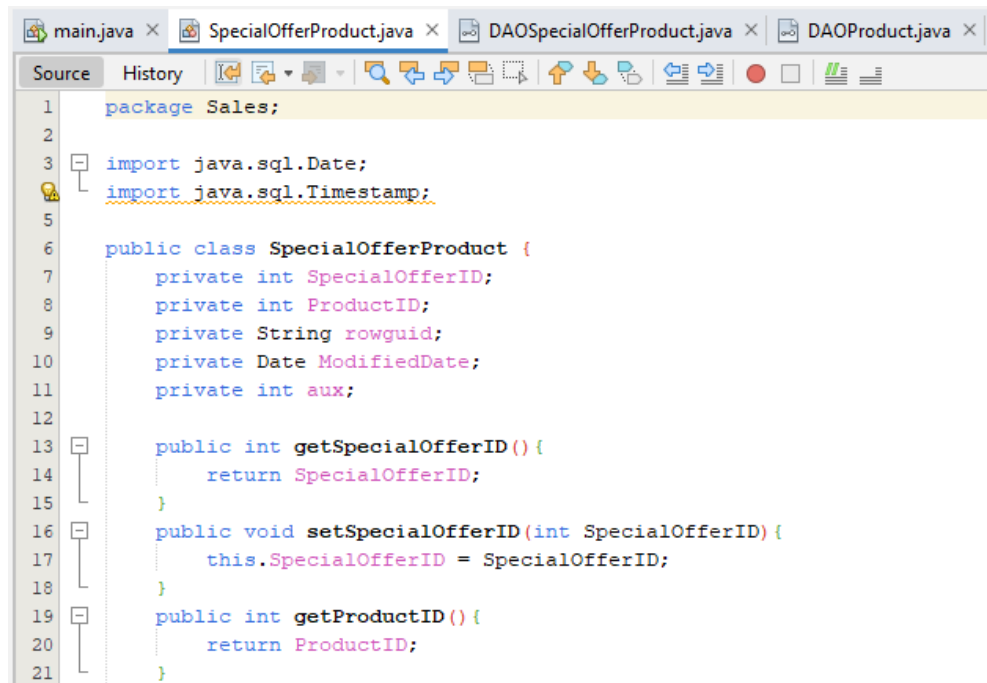
    public void conectar() throws Exception {
        try {
            conexion = DriverManager.getConnection(BD_URL, usuario, contraseña);
            Class.forName(JDBC_DRIVER);

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

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

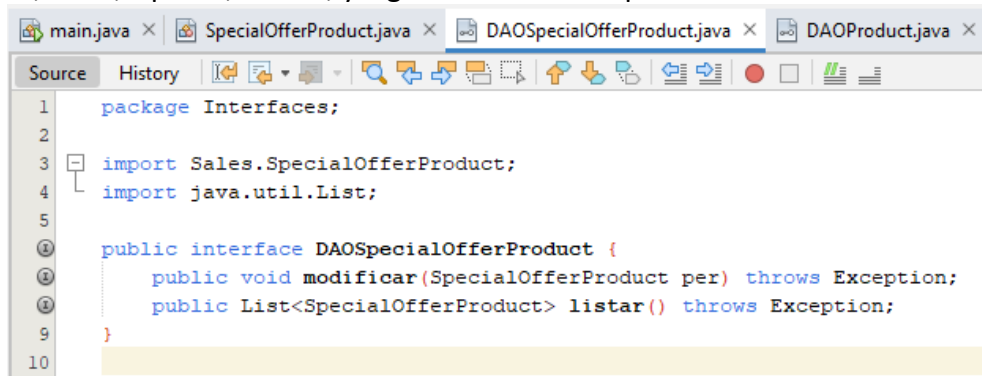

Para implementar el patron DAO, tenemos que:

- 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.

```

public class DAOSpecialOfferProductImpl extends ConexionMySQL implements DAOSpecialOfferProduct {

    @Override
    public void modificar(SpecialOfferProduct per) throws Exception { ...21 lines }

    @Override
    public List<SpecialOfferProduct> listar() throws Exception { ...28 lines }

    @Override
    public List<SpecialOfferProduct> Consulta_5(SpecialOfferProduct per) throws Exception {
        List<SpecialOfferProduct> lista = null;
        try {
            this.conectar();
            PreparedStatement st = this.conexion.prepareStatement("Select * from SpecialOfferProduct as so "
                + "inner join(Select ProductID from Product as p "
                + "inner join(Select *from ProductSubcategory where ProductSubcategory.ProductCategoryID=1)as C "
                + "on p.ProductSubcategoryID=C.ProductSubcategoryID) as psid"
                + " on so.ProductID=psid.ProductID;");

            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:

```

***** Selecciona una opcion *****
1. Sales
2. Product
3. Consultas
0. Salir
Escribe una de las opciones

```

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> Consulta_5(SpecialOfferProduct per) throws Exception {
    List<SpecialOfferProduct> lista = null;
    try {
        this.conectar();
        PreparedStatement st = this.conexion.prepareStatement("Select * from SpecialOfferProduct as so "
            + "inner join(Select ProductID from Product as p "
            + "inner join(Select *from ProductSubcategory where ProductSubcategory.ProductCategoryID=1)as C "
            + "on p.ProductSubcategoryID=C.ProductSubcategoryID) as psid"
            + " on so.ProductID=psid.ProductID;");

        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

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

Aplicación

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

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

```
***** Selecciona una opcion *****
1. Sales
2. Product
3. Consultas
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
5. Sales.SalesPerson
6. Sales.SpecialOffer
Escribe una de las opciones
1
DAO Customer
2. READ
0. SALIR
Escribe una de las opciones
2
Listado TOP 5 Sales.Customer
1 -- 0 -- 934 -- 1 -- AW00000001 -- 2014-09-12
2 -- 0 -- 1028 -- 1 -- AW00000002 -- 2014-09-12
3 -- 0 -- 642 -- 4 -- AW00000003 -- 2014-09-12
4 -- 0 -- 932 -- 4 -- AW00000004 -- 2014-09-12
5 -- 0 -- 1026 -- 4 -- AW00000005 -- 2014-09-12
```

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

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. ProductSubCategory
Escribe una de las opciones
1
DAO Product

2. READ
0. SALIR
Escribe una de las opciones
2
Read
999 -- Road-750 Black, 52 -- BK-R19B-52 -- 343.6496
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
990 -- Mountain-500 Black, 42 -- BK-M18B-42 -- 294.5797
```

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

Opción 3: Consultas

Al seleccionar la opción 3 se despliega un menú en el cual seleccionaremos alguna de las consultas que se tenían que implementar

```
3
1. Consulta 1
2. Consulta 2
3. Consulta 3
4. Consulta 4
5. Consulta 5
6. Consulta 6
7. Consulta 7
8. Consulta 8
9. Consulta 9
10. Consulta 10
Escribe una de las opciones
8
Consulta 8
DAO Product

ProductID      Name      ProductNumber  StandardCost  SellStartDate  SellEndDate
-----
709 -- Mountain Bike Socks, M -- S0-B909-M -- 3.3963 -- 2011-05-31 -- 2012-05-29
710 -- Mountain Bike Socks, L -- S0-B909-L -- 3.3963 -- 2011-05-31 -- 2012-05-29
725 -- LL Road Frame - Red, 44 -- FR-R38R-44 -- 187.1571 -- 2011-05-31 -- 2013-05-29
726 -- LL Road Frame - Red, 48 -- FR-R38R-48 -- 187.1571 -- 2011-05-31 -- 2013-05-29
727 -- LL Road Frame - Red, 52 -- FR-R38R-52 -- 187.1571 -- 2011-05-31 -- 2013-05-29
728 -- LL Road Frame - Red, 58 -- FR-R38R-58 -- 187.1571 -- 2011-05-31 -- 2013-05-29
729 -- LL Road Frame - Red, 60 -- FR-R38R-60 -- 187.1571 -- 2011-05-31 -- 2013-05-29
730 -- LL Road Frame - Red, 62 -- FR-R38R-62 -- 187.1571 -- 2011-05-31 -- 2013-05-29
731 -- ML Road Frame - Red, 44 -- FR-R72R-44 -- 352.1394 -- 2011-05-31 -- 2012-05-29
```

En esta imagen se muestra el resultado de la consulta 8

Consultas;

1. La información de los clientes de almacenarse por región, considerando las regiones de acuerdo con el atributo group de SalesTerritory
2. Listar datos del empleado que atendió mas ordenes por territorio
3. Listar los datos del cliente con mas ordenes solicitadas en la región "North America"
4. Listar el producto mas solicitado en la región "Europe"
5. Listar las ofertas que tienen mas productos de la categoría "Bikes"
6. Listar los 3 productos menos solicitados en la región "Pacific"
7. Actualizar la subcategoria de los productos con ProductID del 1 al 4 a la subcategoria válida para el tipo de producto.
8. Listar los productos que no estén disponibles a la venta.
9. Listar los clientes del territorio 1 al 4 que no tengas asociado un valor en PersonID
10. Listar los clientes del territorio 1 que tengan ordenes en otro territorio.