



# 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

Fragmentacion Horizontal.....	3
Fragmentación propuesta.....	7
Comprobando fragmentación seleccionada .....	7
configuración de servidores vinculados .....	13
Implementación de las consultas.....	18
Implementación de consultas .....	21
Implementación consultas en el DAO.....	21
Aplicación .....	22

## 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

```
SP_1 = SpecialOfferProduct Product
SpecialOfferProduct_F = ( ProductSubCategoryID Between 1 and 3 (Product))
SELECT *FROM Sales.SpecialOfferProduct as sop
where sop.ProductID IN(
Select p.ProductID FROM Production.Product as p
where p.ProductSubcategoryID BETWEEN 1 AND 3)

SP_2 = ProductSubCategory Product
```

```

SP_2 = (ProductSubCategoryID Between 1 and 3 (Product))
SELECT *FROM Production.ProductSubCategory as psc
WHERE ProductID IN{
    Select ProductID
    FROM Production.Product
    Where ProductID BETWEEN 1 AND 4
}

```

```

SP_3 = SpecialOfferProduct Product
SP_3 = (ProductSubCategoryID Between 1 and 3 (Product))
SELECT ProductID, SellStartDate, SellEndDate
FROM Production.Product
where SellEndDate IS NOT NULL

```

## 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, asimismo mencionar que en la consulta 7 y 8 no se vuelve a ocupar estos fragmentos o algún otro que acceda al resto de las categorías. Pues ProductID solo se limita de 1 a 4, y por otro lado el fragmento de productos “Que no están a la venta” solo hace referencia a sí mismo y tampoco se ocupa nuevamente.

### Analizando posible fragmentación por Territorios.

Esta se llevará a cabo por territorio, tomando como base la fragmentación de la tabla Customer, es importante aquí resaltar la relevancia que tiene la fragmentación por territorio debido a que es usado por las consultas 2, 3, 4, 6, 9, 10 siendo de esta manera mas de la mitad de las consultas que recurrirán al uso de estos fragmentos, por lo tanto se tomará esta fragmentación como propuesta y a continuación seguirá siendo analizada.

## Comprobando fragmentación seleccionada

Se tomará en cuenta que el primer enunciado solo se trata de una condición, respetando el numero de enunciado, se considerarán 9 consultas.

### Evaluando predicados

Predicado	Consultas que se relacionan	Porcentaje que representa	Frecuencia de acceso
<b>TerritoryID entre 1 y 6</b>	<b>2,3,9,10</b>	<b>44.44% o 0.4444</b>	<b>2 y 10 una vez al mes</b>

Con base a la información anterior se puede catalogar a este predicado como relevante, debido a su uso en casi la mitad de las consultas y dos de ellas son ejecutadas al menos 2 veces al mes.

Predicado	Consultas que se relacionan	Porcentaje que representa	Frecuencia de acceso
<b>TerritoryID= 7,8 y 10</b>	<b>2,4</b>	<b>22.22% o 0.2222</b>	<b>Se ejecuta 1 vez al mes</b>

En este caso se puede fácilmente apreciar que este predicado sigue siendo relevante debido a que es usado en 2 consultas, 1 de ellas depende de el primer predicado analizado y este mismo manteniendo la frecuencia de acceso.

Predicado	Consultas que se relacionan	Porcentaje que representa	Frecuencia de acceso
<b>TerritoryID= 9</b>	<b>2,6</b>	<b>22.22% o 0.2222</b>	<b>Se ejecuta 1 vez al mes</b>

De igual manera tiene una frecuencia de acceso y consultas relacionadas con valores semejantes al predicado anterior, esto ayuda a equilibrar con qué probabilidad se usarán los fragmentos, la cual de esta manera se mantiene similar.



## Revisando completitud

Por medio de la cláusula Join se verificará que no hay datos sin clasificar podemos ver que la fragmentación de la Tabla Sales.SalesOrderHeader.

```
28
29
30 -- Use Europa
31 -- RECONSTRUCCION
32 Select * From [Europa].[dbo].[SalesOrderHeader]
33 UNION
34 Select * FROM [Norteamérica].[dbo].[SalesOrderHeader]
35 UNION
36 Select * FROM [Pacífico].[dbo].[SalesOrderHeader]
37 GO
38
```

SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber	AccountNumber	CustomerID	SalesPersonID	TerritoryID	BillToAddressID	ShipToAddressID	ShipMethodID
43659	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043659	PO522145767	10-4020-000676	29625	279	5	985	985	5
43660	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043660	PO18850127500	10-4020-000117	29672	279	5	921	921	5
43661	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043661	PO18473108620	10-4020-000442	29734	282	6	517	517	5
43662	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043662	PO18444174044	10-4020-000227	29994	282	6	482	482	5
43663	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043663	PO18009186470	10-4020-000910	29965	276	4	1073	1073	5
43664	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043664	PO18611712183	10-4020-000397	29986	280	1	876	876	5
43665	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043665	PO16588191572	10-4020-000146	29980	283	1	849	849	5
43666	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043666	PO16008173883	10-4020-000511	30052	276	4	1074	1074	5
43667	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043667	PO15426132599	10-4020-000646	29974	277	3	629	629	5
43668	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043668	PO14732103095	10-4020-000514	29614	282	6	529	529	5
43669	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043669	PO14123169936	10-4020-000578	29747	283	1	895	895	5
43670	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043670	PO14384116310	10-4020-000504	29566	275	3	810	810	5
43671	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043671	PO13878119376	10-4020-000200	29690	283	1	855	855	5
43672	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043672	PO13562153037	10-4020-000119	30067	282	6	464	464	5
43673	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043673	PO13775141242	10-4020-000618	29844	275	2	821	821	5
43674	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043674	PO12760141756	10-4020-000083	29596	282	6	458	458	5
43675	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043675	PO12412106464	10-4020-000670	29627	277	3	631	631	5
43676	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043676	PO11861165059	10-4020-000017	29811	275	5	755	755	5
43677	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043677	PO11049174788	10-4020-000679	29624	278	6	556	556	5
43678	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043678	PO10817150168	10-4020-000203	29689	281	4	1021	1021	5
43679	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043679	PO10527142759	10-4020-000480	29761	278	6	525	525	5
43680	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043680	PO10730130087	10-4020-000491	29489	281	4	1069	1069	5
43681	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043681	PO1189177803	10-4020-000423	29661	279	5	955	955	5
43682	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043682	PO1586134200	10-4020-000486	29759	275	2	808	808	5
43683	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043683	PO252113807	10-4020-000508	29487	283	1	889	889	5
43684	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043684	PO3393138842	10-4020-000549	29912	279	5	971	971	5
43685	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043685	PO4178134783	10-4020-000450	30004	279	5	958	958	5
43686	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043686	PO5075125561	10-4020-000344	29606	283	1	871	871	5
43687	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043687	PO4959110629	10-4020-000269	29810	275	2	783	783	5
43688	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043688	PO5360136369	10-4020-000181	29718	275	2	771	771	5
43689	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043689	PO5626159507	10-4020-000166	29646	277	4	576	576	5
43690	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043690	PO6235146326	10-4020-000431	29533	275	3	801	801	5
43691	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043691	PO6499111675	10-4020-000292	29570	277	4	592	592	5
43692	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043692	PO7859167017	10-4020-000221	29962	281	4	1024	1024	5
43693	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043693	PO8120182325	10-4020-000485	29491	275	5	807	807	5
43694	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043694	PO8657102050	10-4020-000315	29549	279	5	943	943	5
43695	8	2011-05-31 00:00:00.000	2011-06-12 00:00:00.000	2011-06-07 00:00:00.000	5	0	5043695	PO10179176559	10-4020-000027	29958	279	5	911	911	5

Tomando en cuenta que en la tabla SalesOrderHeader inicialmente se tienen 31,465 tuplas de datos podemos comprobar que efectivamente al hacer la UNIÓN de los 3 fragmentos obtenemos la tabla original.

Comprobando por medio de Intersección de fragmentos.

```
40
41 -- DISYUNCIÓN
42 Select * From [Europa].[dbo].[SalesOrderHeader]
43 INTERSECT
44 Select * FROM [Norteamérica].[dbo].[SalesOrderHeader]
45 INTERSECT
46 Select * FROM [Pacífico].[dbo].[SalesOrderHeader]
47 GO
48
```

SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber	AccountNumber	CustomerID	SalesPersonID	TerritoryID	BillToAddressID	ShipToAddressID	ShipMethodID	CreditCardID	CreditCardApprovalCode
--------------	----------------	-----------	---------	----------	--------	-----------------	------------------	---------------------	---------------	------------	---------------	-------------	-----------------	-----------------	--------------	--------------	------------------------

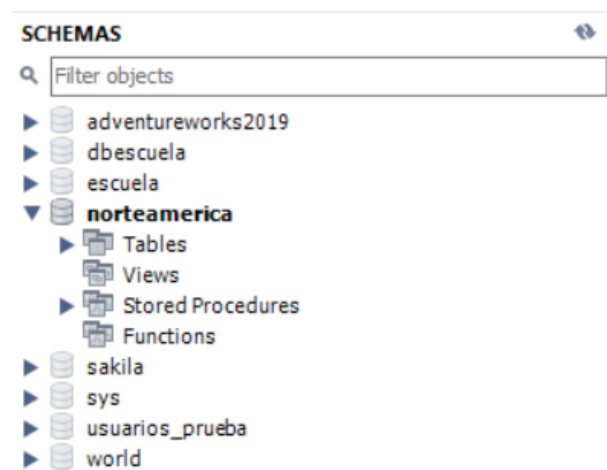
Aquí nos encontramos un conjunto vacío, esto quiere decir que ningún dato se encuentra duplicado en alguna otra tabla, cada tupla de datos es única y cada fragmento posee tuplas distintas que al unirse formarán la tabla original, ya comprobado por medio de la cláusula JOIN

## Asignacion de Fragmentos

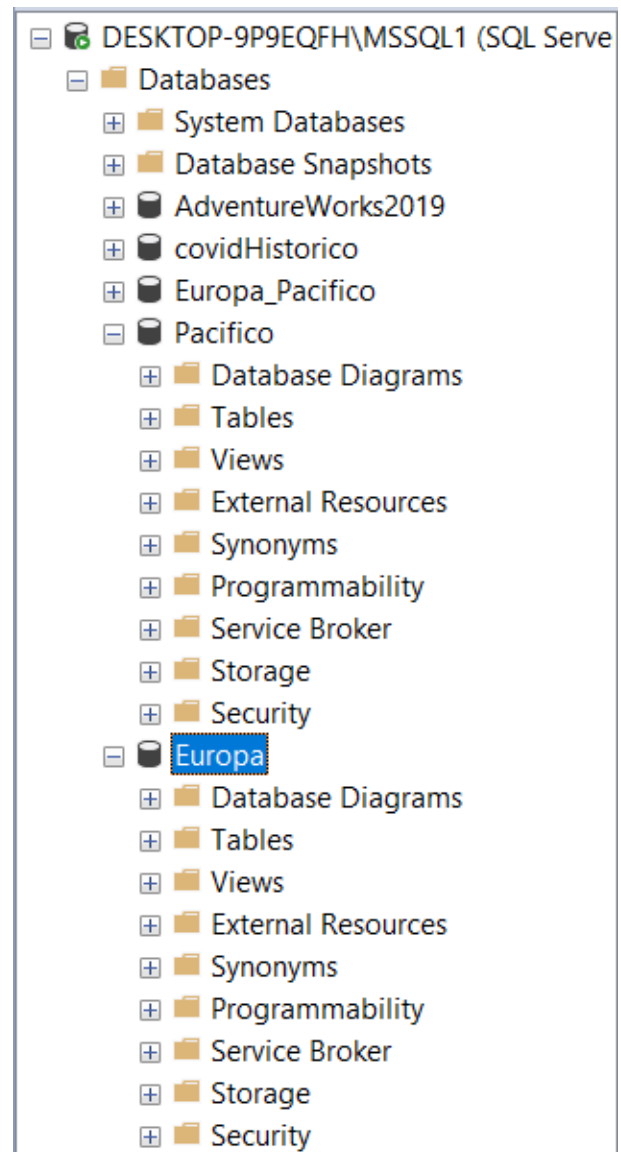
Como se menciona en el apartado anterior, se crean 4 fragmentos dependiendo del Territorio, ahora los fragmentos creados los vamos a alojar en MySQL y SQLServer.

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:

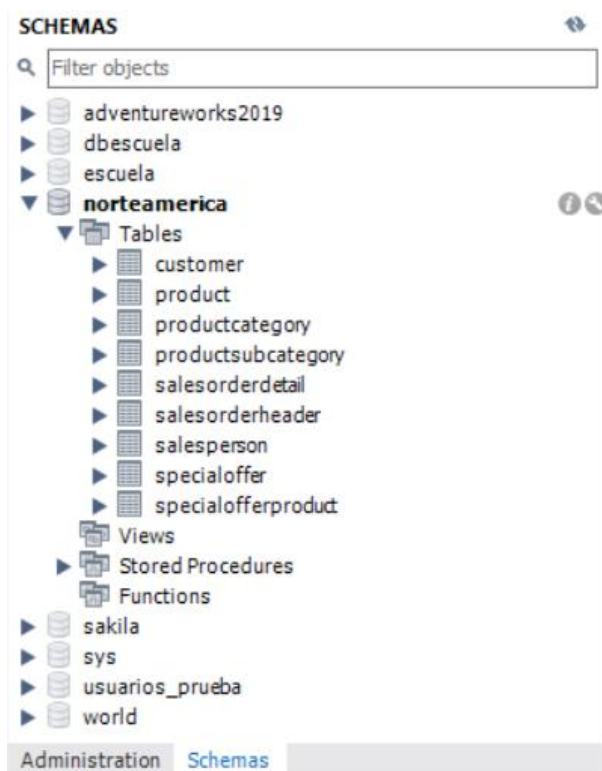
- SpecialOffer
- SpecialOfferProduct
- SalesPerson
- Product
- ProductCategory
- ProductSubcategory

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

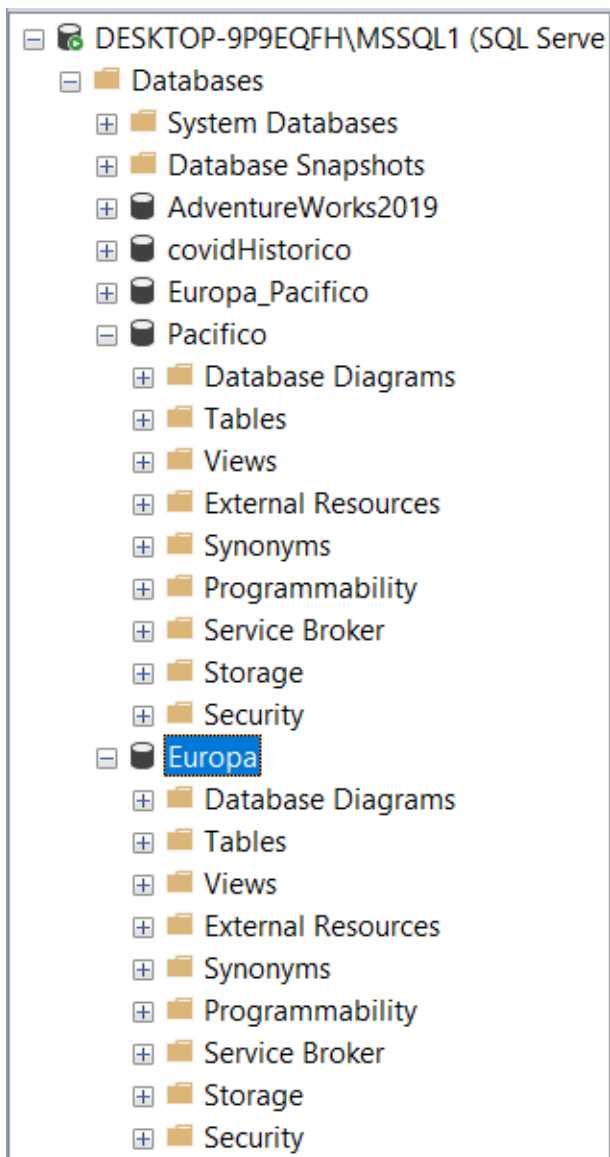
Storage

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.

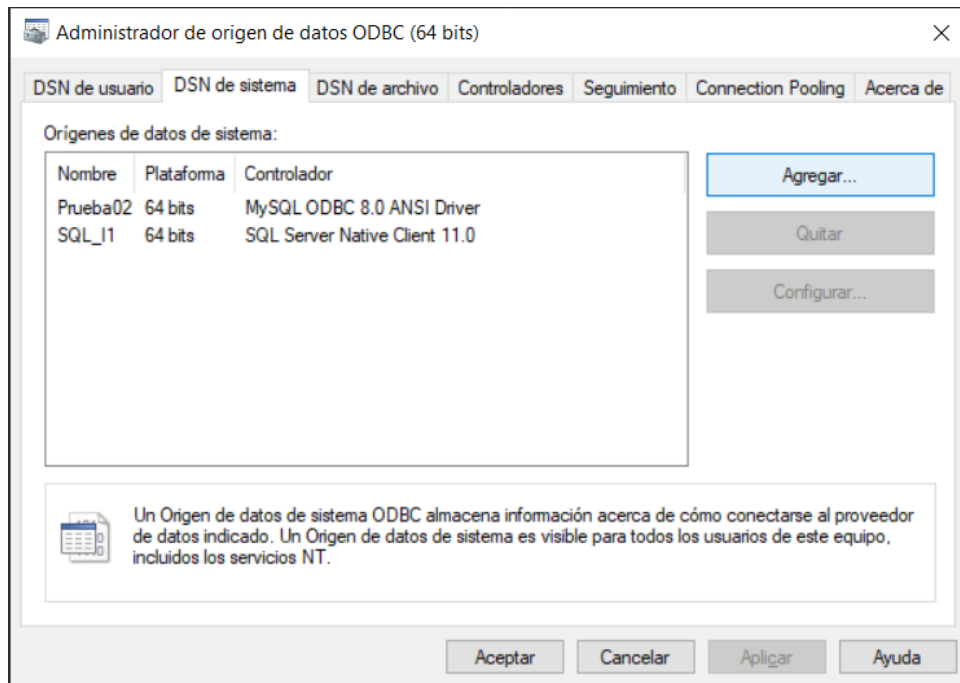


Instancia MySQL el fragmento  
NorteAmerica

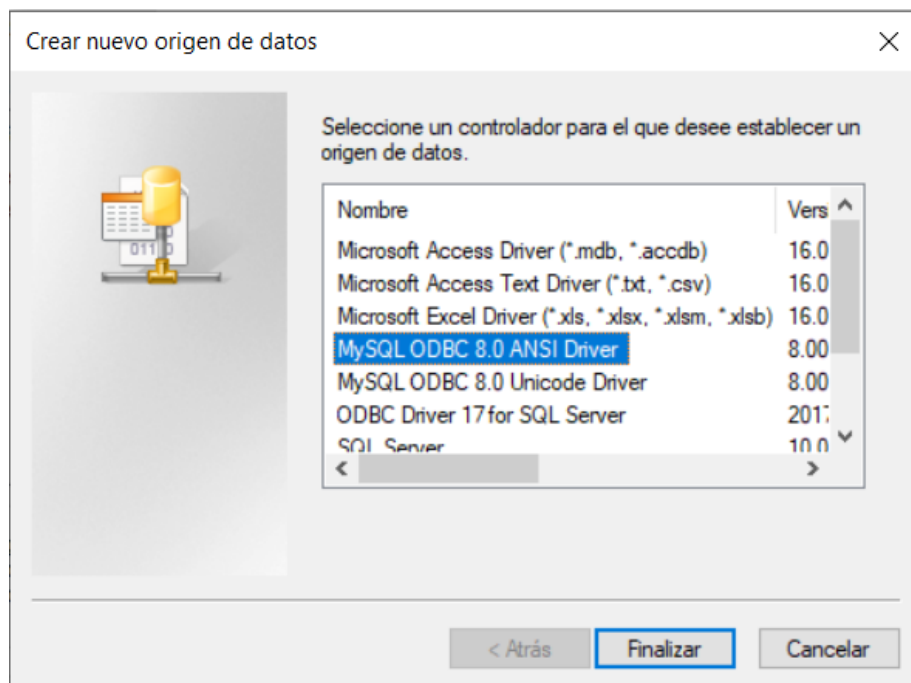
Instancia SQL Server con los fragmentos Europa y Pacífico

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.

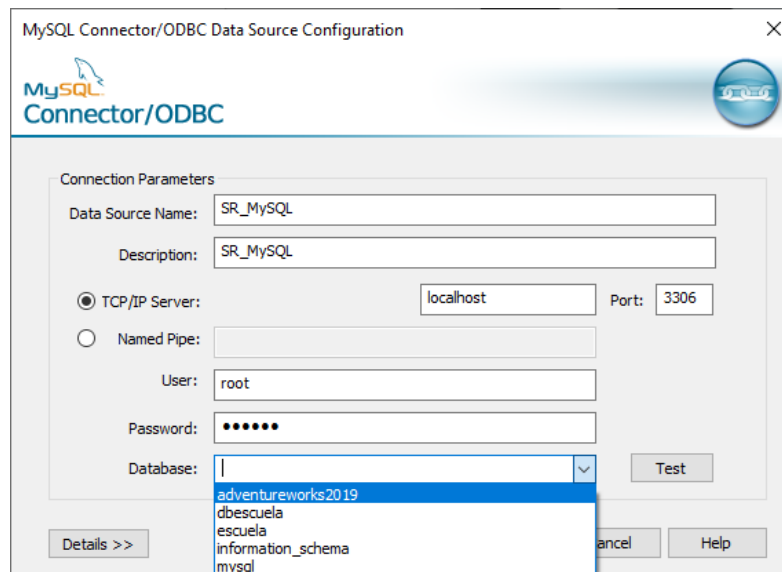
- Agregar un nuevo DSN de sistema en el administrador de origen de datos ODBC



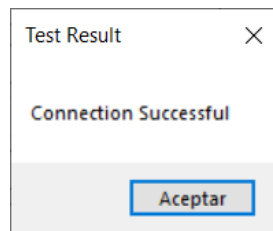
- Como nos vamos a conectar a MySQL, Seleccionamos MySQL ODBC 8.0 ---



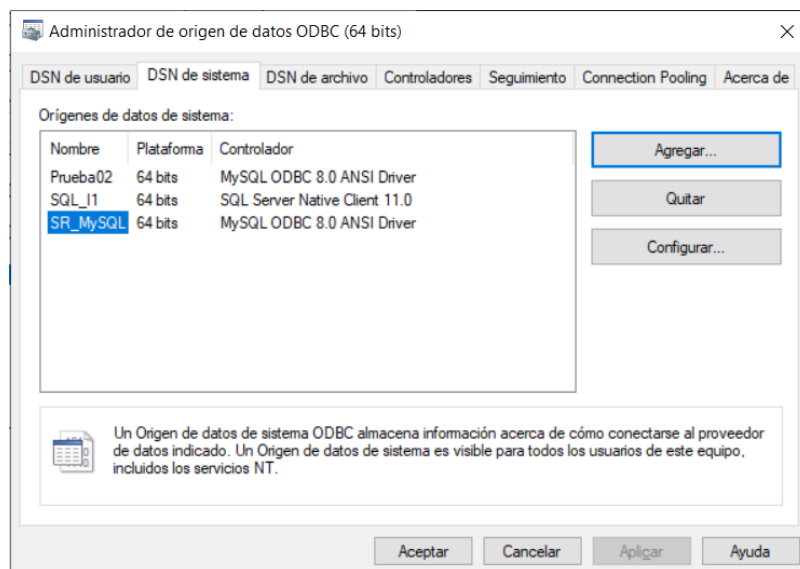
- En esta ventana ponemos el nombre y los datos para la conexión a MySQL, también seleccionamos la base de datos a acceder



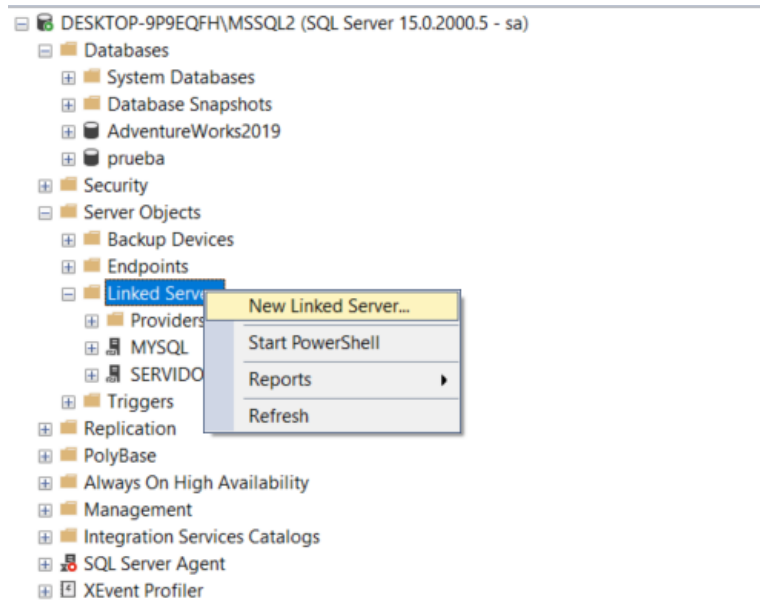
- Probamos que la conexión sea exitosa



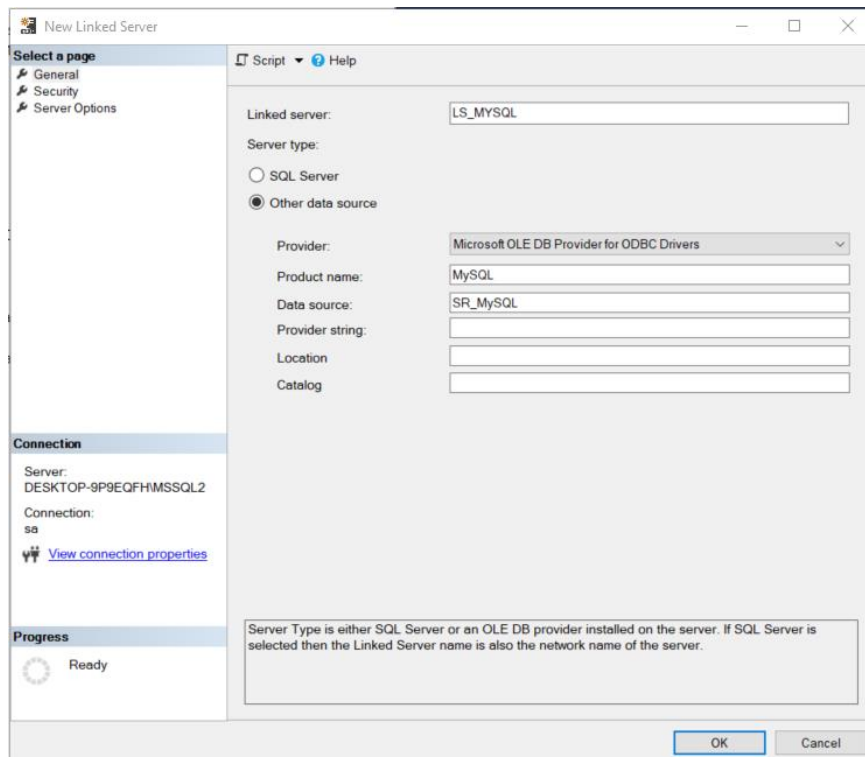
- Ya tenemos el nuevo DSN de sistema agregado



Ahora tenemos que crear el servidor vinculado en SQL Server

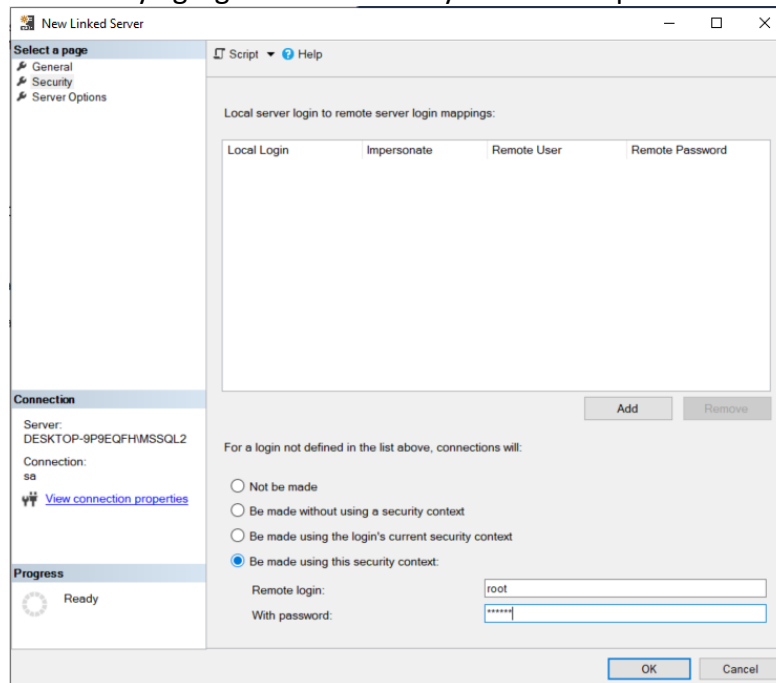


- Agregamos el nombre del servidor remoto
- Seleccionamos el proveedor para ODBC Drivers
- En el campo Data Source agregamos el nombre del DSN que creamos previamente





- 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')

```

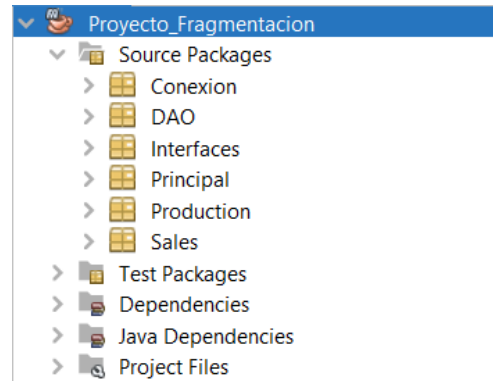
119 %

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;

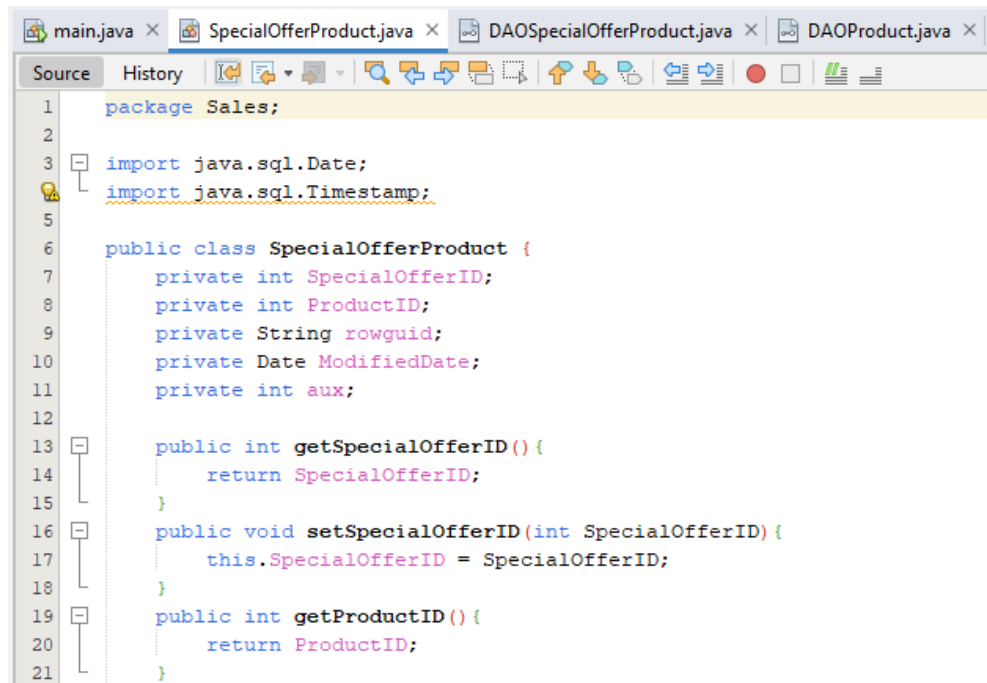
    public void conectar() throws Exception {
        try {
            conexion = DriverManager.getConnection(BD_URL, usuario, contraseña);
            Class.forName(JDBC_DRIVER);

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

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

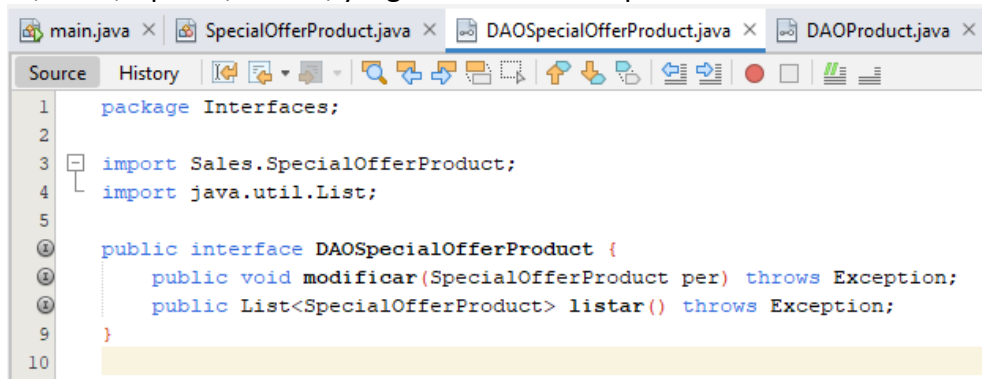
Para implementar el patron DAO, tenemos que:

- 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 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.