Equipo 2

Optimización de consultas

Consultas Propuestas

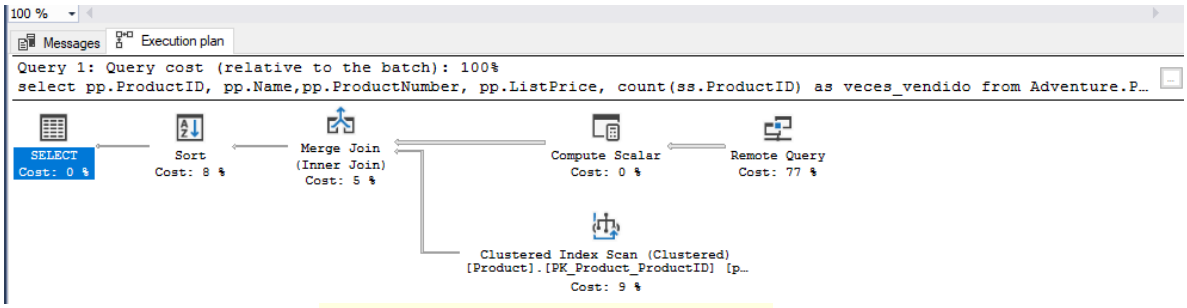
1. Listar los productos más vendidos con un precio menor o igual a 100 y que el color sea amarillo.
2. Listar los productos más vendidos con un precio mayor o igual a 600 y un nivel de cuidado del stock igual a 500.
3. Listar los productos más vendidos con un precio estándar menor o igual a 250 y el dia de manufactura sea igual a 1.
4. Determinar el número de ventas en el territorio 3
5. Determinar el número de ventas donde el total de la compra sea mayor a 0 y menor a 1000
6. Determinar el número de ventas donde el total de la compra sea mayor a 1000 y menor a 2000
7. Determinar el número de ventas donde el total de la compra sea mayor a 2000
8. Determinar las ventas donde los impuestos aplicados sean mayor o igual a los 500 y el territorio sea el 3
9. Listar las órdenes del territorio 9 con fecha de modificación del 2011-06-30 00:00:00.000.
10. Actualizar el nivel de existencias de seguridad y el color (SafetyStockLevel) del producto indicado, asi como la fecha de modificacion.
11. Registrar un nuevo producto.
12. Actualizar el descuento del detalle de la orden de compra y el ID de oferta.

Consulta 1

1. Listar los productos más vendidos con un precio menor o igual a 100 y que el color sea amarillo.

```
select pp.ProductID, pp.Name,pp.ProductNumber, pp.ListPrice,
              count(ss.ProductID) as veces_vendido
              from Adventure.Production.Product pp
              inner join SQLS2.AdventureSales.Sales.SalesOrderDetail ss
              on pp.ProductID = ss.ProductID
              where pp.ListPrice <=100 and pp.Color = 'Yellow'
              group by ss.ProductID, pp.Name, pp.ProductID,pp.ProductNumber,
pp.ListPrice
              order by count(ss.ProductID) desc;
```

   Se puede observar en esta primera consulta que el mayor peso de la ejecución se lleva a cabo en el servidor AdventureSales



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0127253 (9%)
Estimated I/O Cost	0.0120139
Estimated Subtree Cost	0.0127253
Estimated CPU Cost	0.0007114
Estimated Number of Executions	1
Estimated Number of Rows to be Read	504
Estimated Number of Rows for All Executions	27.3078
Estimated Number of Rows Per Execution	27.3078
Estimated Row Size	107 B
Ordered	True
Node ID	5

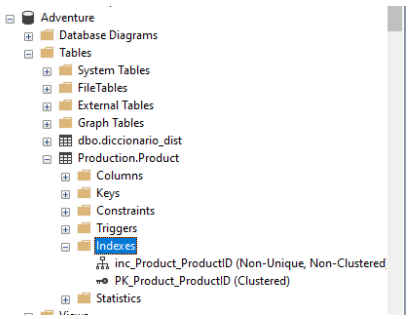
Predicate
[Adventure].[Production].[Product].[ListPrice] as [pp].[ListPrice] <= ($100.0000) AND [Adventure].[Production].[Product].[Color] as [pp].[Color]=N'Yellow'

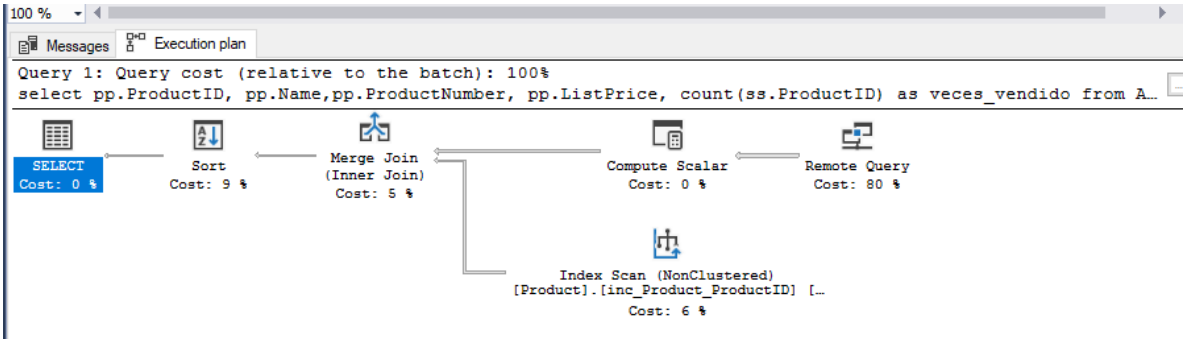
Object
[Adventure].[Production].[Product].[PK_Product_ProductID] [pp]

Output List
[Adventure].[Production].[Product].ProductID, [Adventure].[Production].[Product].Name, [Adventure].[Production].[Product].ProductNumber, [Adventure].[Production].[Product].ListPrice

Creando un índice no agrupado por medio del campo ProductID e incluyendo los campos de las columnas que se utilizaran en las consultas posteriores se obtiene lo siguiente:

```
create NONCLUSTERED INDEX inc_Product_ProductID
ON Production.Product (ProductID) include
(Name , ProductNumber, ListPrice, Color,
SafetyStockLevel,StandardCost,DaysToManufacture);
GO
```





Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0082808 (6%)
Estimated I/O Cost	0.0075694
Estimated Subtree Cost	0.0082808
Estimated CPU Cost	0.0007114
Estimated Number of Executions	1
Estimated Number of Rows to be Read	504
Estimated Number of Rows for All Executions	27.3078
Estimated Number of Rows Per Execution	27.3078
Estimated Row Size	107 B
Ordered	True
Node ID	5

Predicate

[Adventure].[Production].[Product].[ListPrice] as [pp].[ListPrice]<= ($100.0000) AND [Adventure].[Production].[Product].[Color] as [pp].[Color]=N'Yellow'

Object

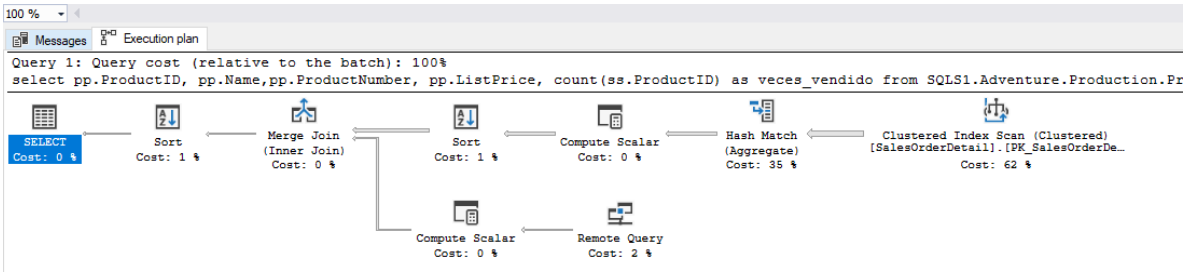
[Adventure].[Production].[Product].[inc_Product_ProductID] [pp]

Output List

[Adventure].[Production].[Product].ProductID, [Adventure].[Production].[Product].Name, [Adventure].[Production].[Product].ProductNumber, [Adventure].[Production].[Product].ListPrice

Se puede observar que ejecutando la consulta desde el primer servidor donde se tiene la tabla producto, se aprecia que el costo estimado de la consulta en la tabla Product es de .01 (6%) y el cual se redujo por medio del índice no agrupado a .008 (4%) sin embargo la apreciación es mínima comparándola con el índice agrupado en la llave primaria de la tabla antes mencionada.

Por otro lado se analiza la consulta desde el servidor 2 donde se encuentra la mayor carga de trabajo, obteniendo lo siguiente:



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated I/O Cost	0.915718
Estimated Operator Cost	1.04932 (62%)
Estimated CPU Cost	0.133606
Estimated Subtree Cost	1.04932
Estimated Number of Executions	1
Estimated Number of Rows for All Executions	121317
Estimated Number of Rows Per Execution	121317
Estimated Number of Rows to be Read	121317
Estimated Row Size	11 B
Ordered	False
Node ID	6

Object

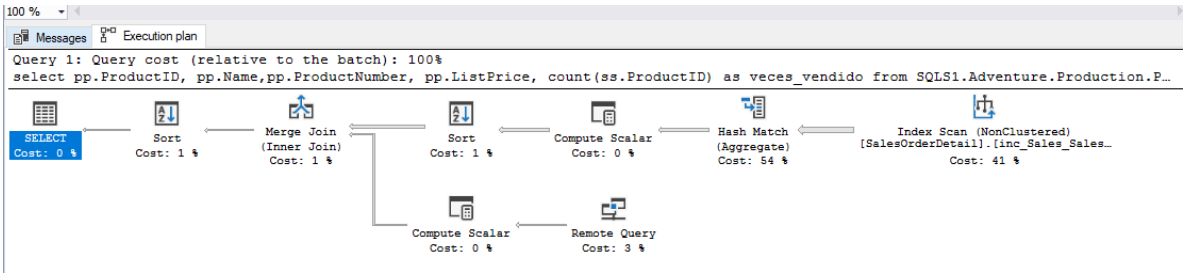
[AdventureSales].[Sales].[SalesOrderDetail].[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] [ss]

Output List

[AdventureSales].[Sales].[SalesOrderDetail].ProductID

Creando un índice no agrupado en la tabla SalesOrderDetail

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderDetail
ON Sales.SalesOrderDetail (SalesOrderID) include (ProductID,OrderQty, UnitPrice);
```



Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.447101 (41%)
Estimated I/O Cost	0.313495
Estimated Subtree Cost	0.447101
Estimated CPU Cost	0.133606
Estimated Number of Executions	1
Estimated Number of Rows to be Read	121317
Estimated Number of Rows for All Executions	121317
Estimated Number of Rows Per Execution	121317
Estimated Row Size	11 B
Ordered	False
Node ID	6

Object
[AdventureSales].[Sales].[SalesOrderDetail].
[inc_Sales_SalesOrderDetail] [ss]

Output List
[AdventureSales].[Sales].[SalesOrderDetail].ProductID

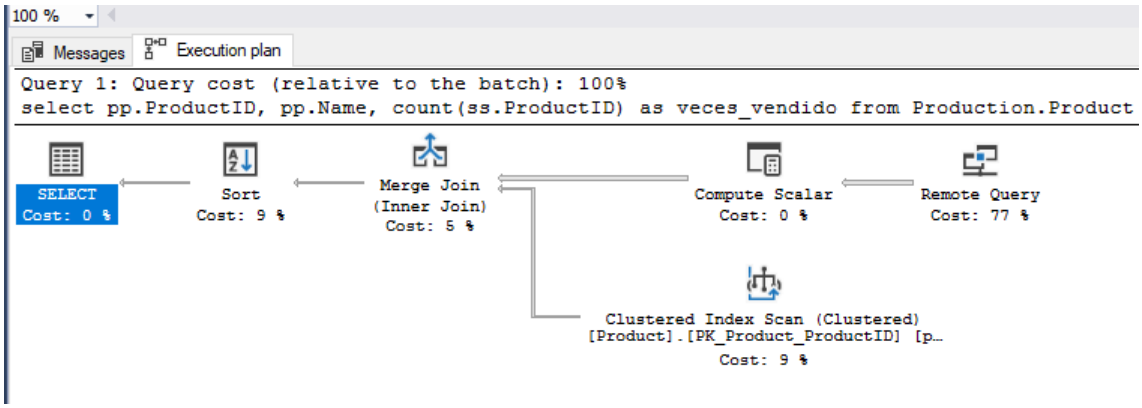
Podemos observar que en este caso el índice no agrupado disminuyo el coste estimado de 0.9 (62%) a 0.44 (41%).

Consulta 2

2. Listar los productos más vendidos con un precio mayor o igual a 600 y un nivel de cuidado del stock igual a 500.

```
select pp.ProductID, pp.Name,pp.ProductNumber, pp.ListPrice,
                        count(ss.ProductID) as veces_vendido
                        from SQLS2.Adventure.Production.Product pp
                        inner join Sales.SalesOrderDetail ss
                        on pp.ProductID = ss.ProductID
                        where pp.ListPrice <=100 and pp.Color = 'Yellow'
                        group by ss.ProductID, pp.Name,
pp.ProductID,pp.ProductNumber, pp.ListPrice
                        order by count(ss.ProductID) desc;
```

   Al igual que la consulta anterior se utilizaron los mismos índices que en la consulta previa, ya que los campos que usa esta consulta son similares, en la siguientes imágenes se puede observar el análisis desde ambos servidores ejecutando la consulta distribuido.
   Consulta ejecutada desde el servidor 1:



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0127253 (9%)
Estimated I/O Cost	0.0120139
Estimated Subtree Cost	0.0127253
Estimated CPU Cost	0.0007114
Estimated Number of Executions	1
Estimated Number of Rows to be Read	504
Estimated Number of Rows for All Executions	64.4705
Estimated Number of Rows Per Execution	64.4705
Estimated Row Size	75 B
Ordered	True
Node ID	5

Predicate

[Adventure].[Production].[Product].[SafetyStockLevel] as [pp].[SafetyStockLevel]=(500) AND [Adventure].[Production].[Product].[ListPrice] as [pp].[ListPrice]>=($600.0000)

Object

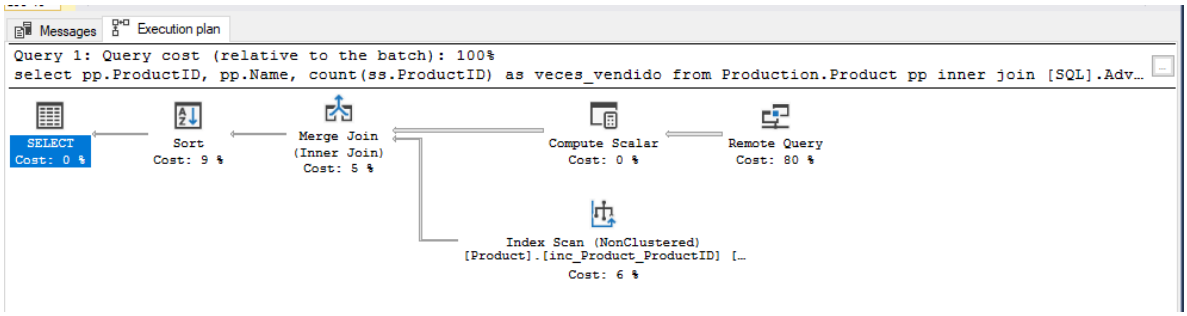
[Adventure].[Production].[Product].[PK_Product_ProductID] [pp]

Output List

[Adventure].[Production].[Product].ProductID, [Adventure].[Production].[Product].Name

Índice no agrupado

```
create NONCLUSTERED INDEX inc_Product_ProductID
ON Production.Product (ProductID) include
(Name , ProductNumber, ListPrice, Color,
SafetyStockLevel,StandardCost,DaysToManufacture);
GO
```



Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0082808 (6%)
Estimated I/O Cost	0.0075694
Estimated Subtree Cost	0.0082808
Estimated CPU Cost	0.0007114
Estimated Number of Executions	1
Estimated Number of Rows to be Read	504
Estimated Number of Rows for All Executions	64.4705
Estimated Number of Rows Per Execution	64.4705
Estimated Row Size	75 B
Ordered	True
Node ID	5

Predicate

[Adventure].[Production].[Product].[SafetyStockLevel] as [pp].[SafetyStockLevel]=(500) AND [Adventure].[Production].[Product].[ListPrice] as [pp].[ListPrice]>=($600.0000)

Object

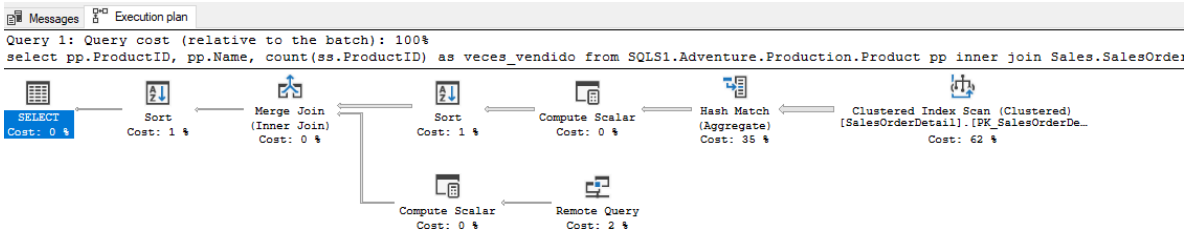
[Adventure].[Production].[Product].[inc_Product_ProductID] [pp]

Output List

[Adventure].[Production].[Product].ProductID, [Adventure].[Production].[Product].Name

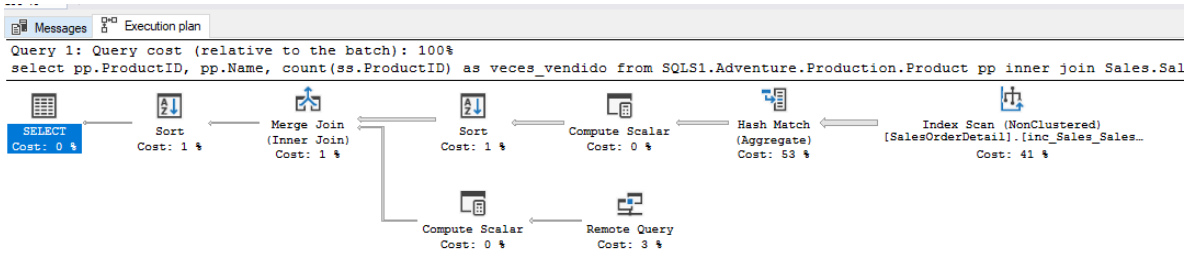
De igual manera se aprecia que el índice no agrupado es un poco más eficiente que el índice agrupado en la llave primaria sin embargo sigue siendo poca la diferencia.

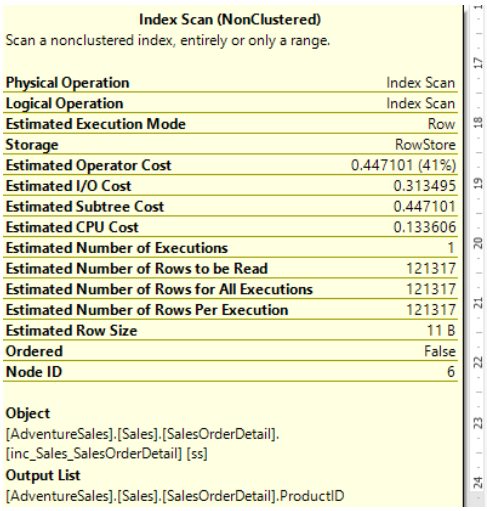
Analizando la consulta desde la instancia 2 se tiene:



Índice no agrupado en tabla SalesOrderDetail:

```
 create NONCLUSTERED INDEX inc_Sales_SalesOrderDetail
ON Sales.SalesOrderDetail (SalesOrderID) include (ProductID,OrderQty, UnitPrice);
```



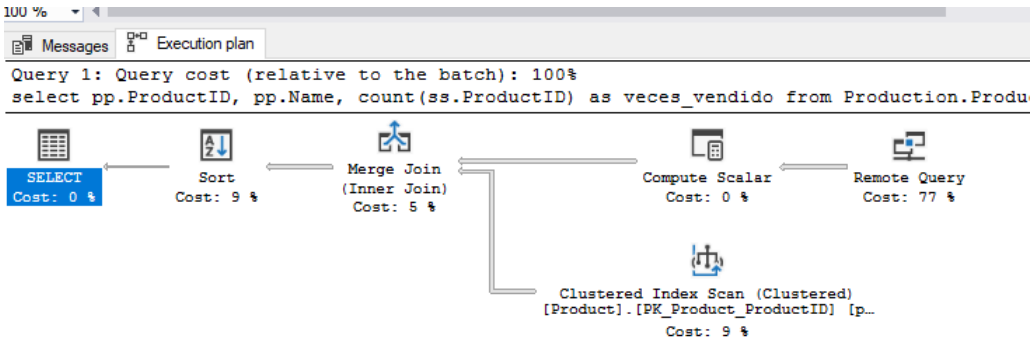


La consulta toma el índice no agrupado ya que es más rápido el acceso a los datos a diferencia del índice agrupado en la llave primaria mostrado previamente.

Consulta 3

```
select pp.ProductID, pp.Name, count(ss.ProductID) as veces_vendido
            from Production.Product pp
            inner join SQLS2.AdventureSales.Sales.SalesOrderDetail ss
            on pp.ProductID = ss.ProductID
            where pp.StandardCost <= 250 and pp.DaysToManufacture = 1
            group by ss.ProductID, pp.Name, pp.ProductID
            order by count(ss.ProductID) asc;
```

3. Listar los productos más vendidos con un precio estándar menor o igual a 250 y el día de manufactura sea igual a 1.
   En las consultas previas pudimos observar la diferencia entre los índices agrupados y no agrupados y la diferencia que hacen en la ejecución de la consulta, para la consulta 3 es un caso similar ya que se utilizan campos semejantes a las consultas previas por tanto se esperaría un resultado similar:
   Consulta ejecutada en el servidor 1:



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0127253 (9%)
Estimated I/O Cost	0.012013
Estimated Subtree Cost	0.012725
Estimated CPU Cost	0.000711
Estimated Number of Executions	1
Estimated Number of Rows to be Read	504
Estimated Number of Rows for All Executions	131.59
Estimated Number of Rows Per Execution	131.59
Estimated Row Size	77 B
Ordered	True
Node ID	5

Predicate

[Adventure].[Production].[Product].[DaysToManufacture] as [pp].[DaysToManufacture]=(1) AND [Adventure].[Production].[Product].[StandardCost] as [pp].[StandardCost]<=($250.0000)

Object

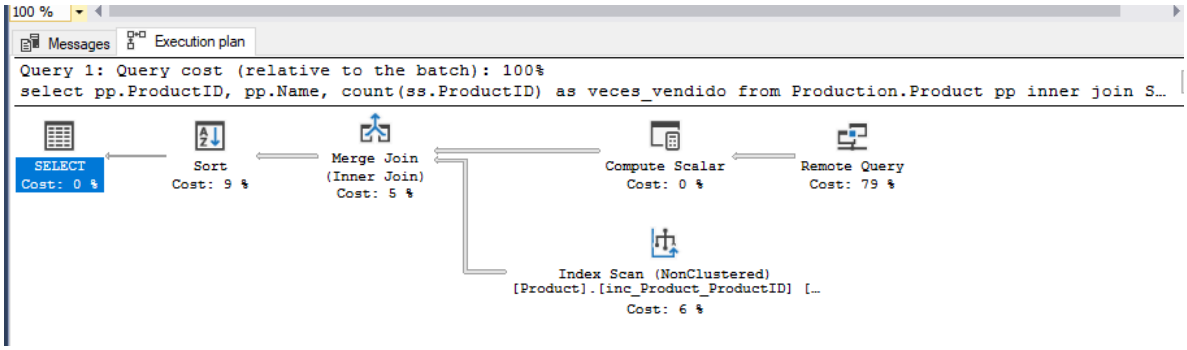
[Adventure].[Production].[Product].[PK_Product_ProductID] [pp]

Output List

[Adventure].[Production].[Product].ProductID, [Adventure].[Production].[Product].Name

Índice no agrupado

```
create NONCLUSTERED INDEX inc_Product_ProductID
ON Production.Product (ProductID) include
(Name , ProductNumber, ListPrice, Color,
SafetyStockLevel,StandardCost,DaysToManufacture);
    GO
```



Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0082808 (6%)
Estimated I/O Cost	0.0075694
Estimated Subtree Cost	0.0082808
Estimated CPU Cost	0.0007114
Estimated Number of Executions	1
Estimated Number of Rows to be Read	504
Estimated Number of Rows for All Executions	131.592
Estimated Number of Rows Per Execution	131.592
Estimated Row Size	77 B
Ordered	True
Node ID	5

Predicate

[Adventure].[Production].[Product].[DaysToManufacture] as [pp].[DaysToManufacture]=(1) AND [Adventure].[Production].[Product].[StandardCost] as [pp].[StandardCost]<=($250.0000)

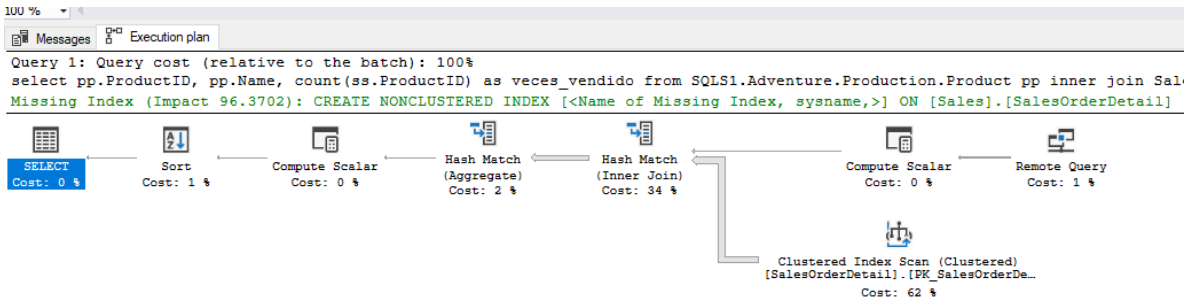
Object

[Adventure].[Production].[Product].[inc_Product_ProductID] [pp]

Output List

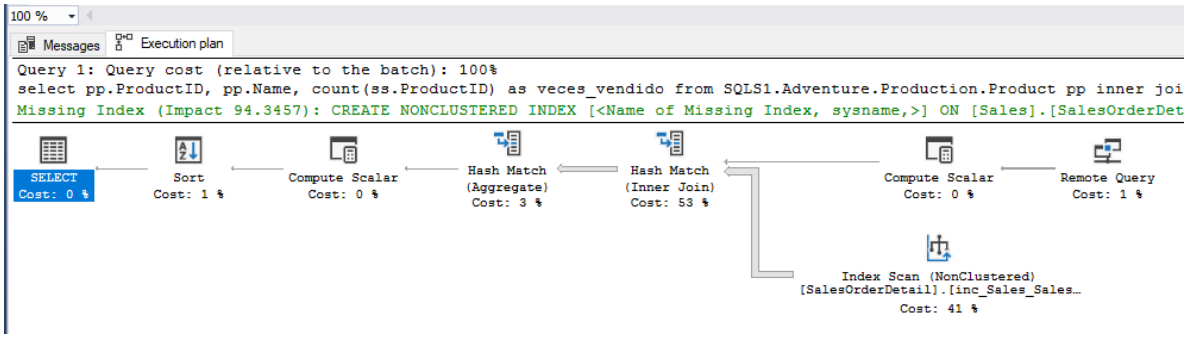
[Adventure].[Production].[Product].ProductID, [Adventure].[Production].[Product].Name

Consulta ejecutada en el servidor 2:



Índice no agrupado en la tabla SalesOrderDetail:

```
Create NONCLUSTERED INDEX inc_Sales_SalesOrderDetail
 ON Sales.SalesOrderDetail (SalesOrderID) include (ProductID,OrderQty, UnitPrice);
```

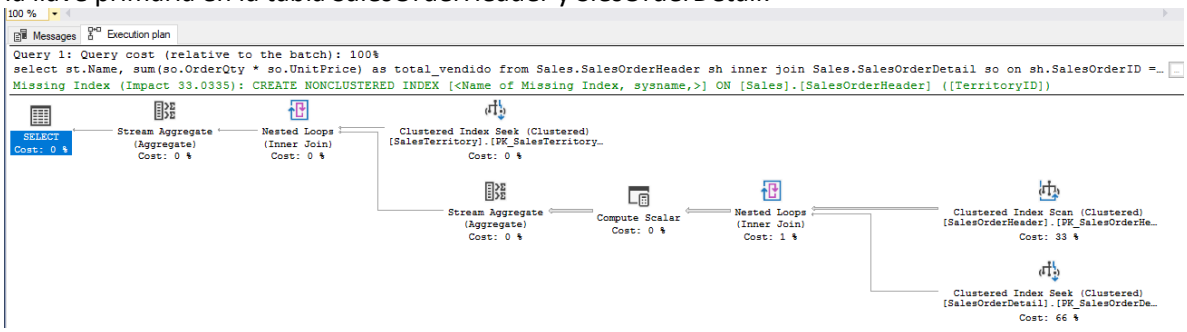


Consulta 4

4. Determinar el total vendido de las ventas en el territorio 3.

```
select st.Name, sum(so.OrderQty * so.UnitPrice) as total_vendido
                    from Sales.SalesOrderHeader sh
                    inner join Sales.SalesOrderDetail so
                    on sh.SalesOrderID = so.SalesOrderID
                    inner join Sales.SalesTerritory st
                    on st.TerritoryID = sh.TerritoryID
                    where st.TerritoryID = 3
                           group by st.TerritoryID, st.Name;
```

Se observa en la siguiente imagen, que se cuenta con dos índices agrupados por medio de la llave primaria en la tabla SalesOrderHeader y SlesOrderDetail:



Clustered Index Seek (Clustered)

Scanning a particular range of rows from a clustered index.

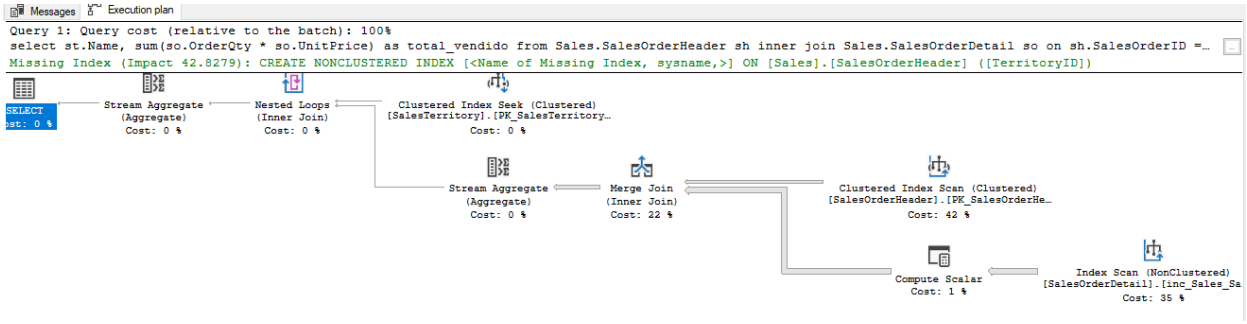
Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	1.0937 (66%)
Estimated I/O Cost	0.003125
Estimated Subtree Cost	1.0937
Estimated CPU Cost	0.0001616
Estimated Number of Executions	385
Estimated Number of Rows to be Read	4.21226
Estimated Number of Rows for All Executions	1621.7201
Estimated Number of Rows Per Execution	4.21226
Estimated Row Size	17 B
Ordered	True
Node ID	8

Object
[AdventureSales].[Sales].[SalesOrderDetail].
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] [so]
Output List
[AdventureSales].[Sales].[SalesOrderDetail].OrderQty, [AdventureSales].
[Sales].[SalesOrderDetail].UnitPrice
Seek Predicates
Seek Keys[1]: Prefix: [AdventureSales].[Sales].
[SalesOrderDetail].SalesOrderID = Scalar Operator([AdventureSales].
[Sales].[SalesOrderHeader].[SalesOrderID] as [sh].[SalesOrderID])

Para eficientar el acceso a la consulta se crearan dos índices en las tablas SalesOrderDetail y SalesOrderHeader:

Creando Índice en SalesOrderDetail

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderDetail
ON Sales.SalesOrderDetail (SalesOrderID) include (ProductID,OrderQty, UnitPrice);
```



Clustered Index Scan (Clustered)

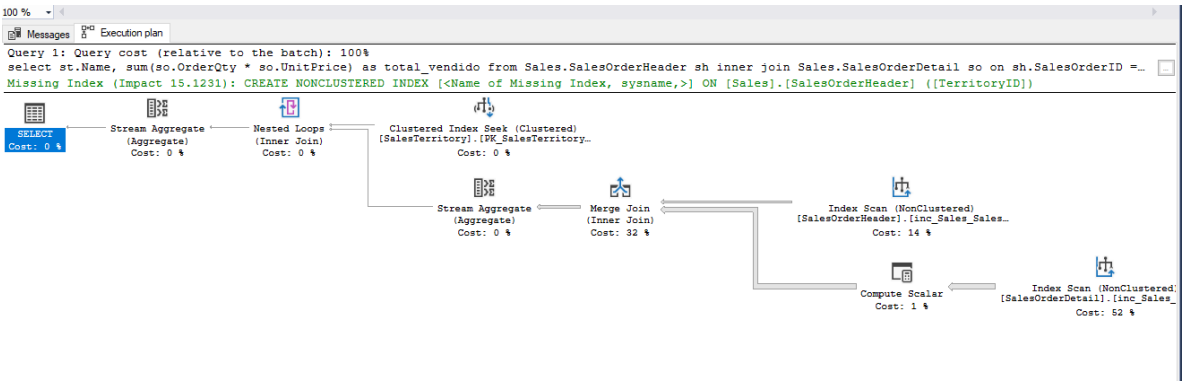
Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.542338 (42%)
Estimated I/O Cost	0.507569
Estimated Subtree Cost	0.542338
Estimated CPU Cost	0.0347685
Estimated Number of Executions	1
Estimated Number of Rows to be Read	31465
Estimated Number of Rows for All Executions	385
Estimated Number of Rows Per Execution	385
Estimated Row Size	15 B
Ordered	True
Node ID	5

Predicate
[AdventureSales].[Sales].[SalesOrderHeader].[TerritoryID] as [sh].
[TerritoryID]=(3)
Object
[AdventureSales].[Sales].[SalesOrderHeader].
[PK_SalesOrderHeader_SalesOrderID] [sh]
Output List
[AdventureSales].[Sales].[SalesOrderHeader].SalesOrderID

creando Índice en la tabla Salen sOrderHeader

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderHeader
ON Sales.SalesOrderHeader(SalesOrderID) include (TerritoryID,DueDate,TaxAmt);
```



Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.123819 (14%)
Estimated I/O Cost	0.0890509
Estimated Subtree Cost	0.123819
Estimated CPU Cost	0.0347685
Estimated Number of Executions	1
Estimated Number of Rows to be Read	31465
Estimated Number of Rows for All Executions	385
Estimated Number of Rows Per Execution	385
Estimated Row Size	15 B
Ordered	True
Node ID	5

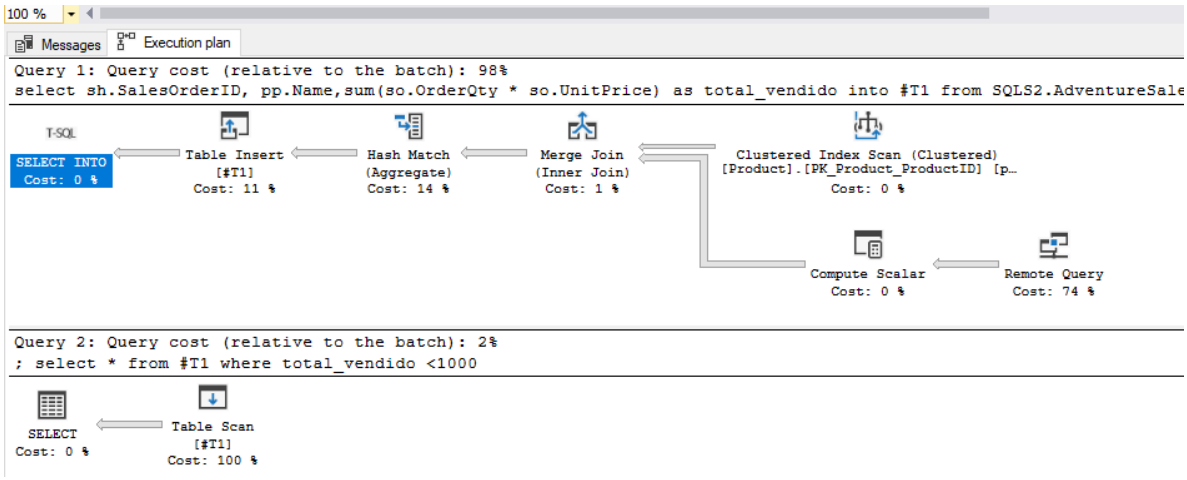
Predicate
[AdventureSales].[Sales].[SalesOrderHeader].[TerritoryID] as [sh].[TerritoryID]=(3)
Object
[AdventureSales].[Sales].[SalesOrderHeader].[inc_Sales_SalesOrderHeader] [sh]
Output List
[AdventureSales].[Sales].[SalesOrderHeader].SalesOrderID

Despues de crear ambos indices pudimos observar que Nasted Loop se elimino y en su lugar se implemento un Merge Join el cual es una de las opciones mas eficientes para reunir datos entre tablas y lo cual

Consulta 5

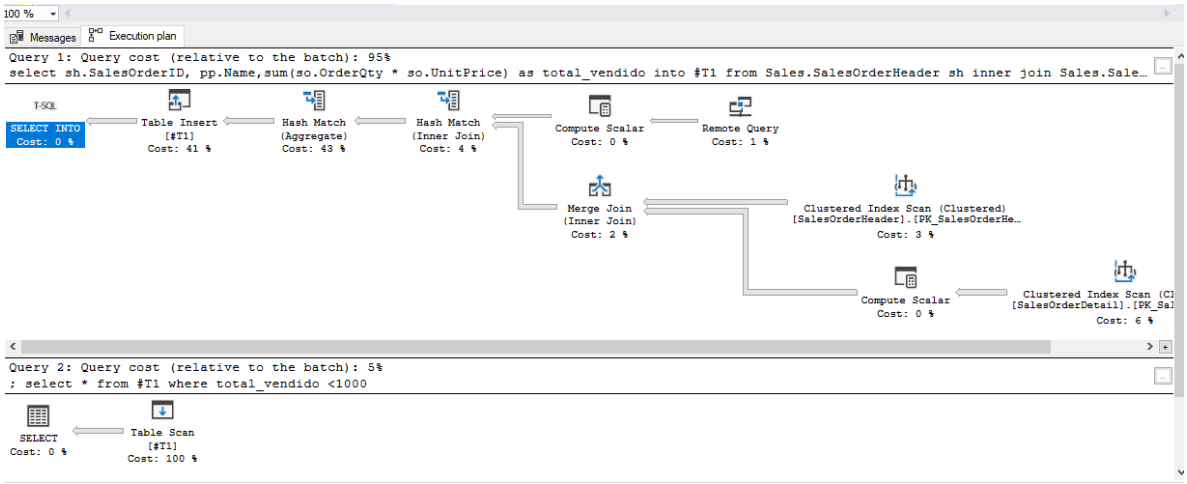
5. Determinar el número de ventas donde el total de la compra sea mayor a 0 y menor a 1000

```
select sh.SalesOrderID,  pp.Name,sum(so.OrderQty * so.UnitPrice) as
total_vendido into #T1
from SQLS2.AdventureSales.Sales.SalesOrderHeader sh
inner join SQLS2.AdventureSales.Sales.SalesOrderDetail so
on sh.SalesOrderID = so.SalesOrderID
inner join Production.Product pp
on pp.ProductID = so.ProductID
group by sh.SalesOrderID, pp.Name
order by sum(so.OrderQty * so.UnitPrice) asc;

    select * from #T1
       where total_vendido <1000;
```



Se puede apreciar que aunque se cuenta con la tabla producto el mayor porcentaje de acceso a la consulta se encuentra en el servidor 2 por tanto se analizó la consulta ejecutándola en la segunda instancia.

Servidor 2:



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

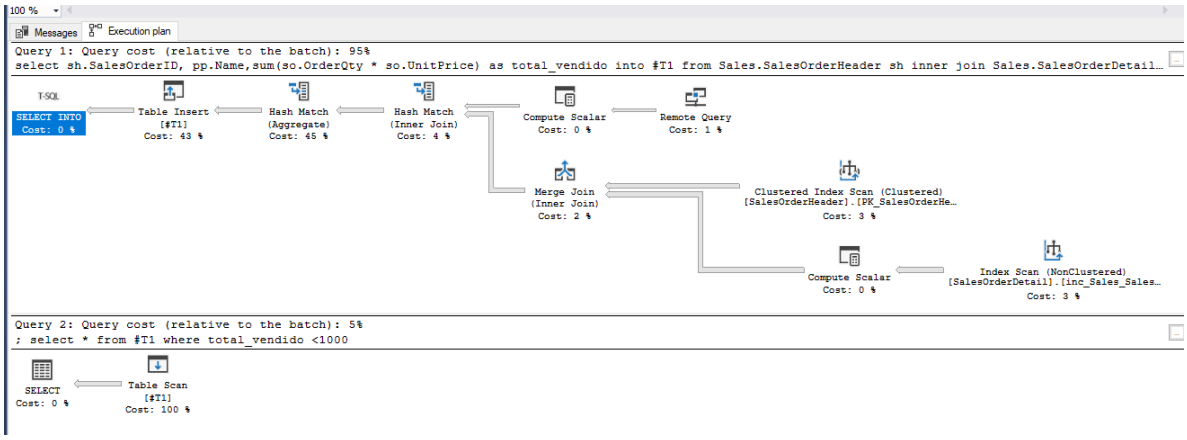
Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	1.04932 (6%)
Estimated I/O Cost	0.915718
Estimated Subtree Cost	1.04932
Estimated CPU Cost	0.133606
Estimated Number of Executions	1
Estimated Number of Rows to be Read	121317
Estimated Number of Rows for All Executions	121317
Estimated Number of Rows Per Execution	121317
Estimated Row Size	25 B
Ordered	True
Node ID	8

Object
[AdventureSales].[Sales].[SalesOrderDetail].
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] [so]

Output List
[AdventureSales].[Sales].[SalesOrderDetail].SalesOrderID,
[AdventureSales].[Sales].[SalesOrderDetail].OrderQty, [AdventureSales].
[Sales].[SalesOrderDetail].ProductID, [AdventureSales].[Sales].
[SalesOrderDetail].UnitPrice

Indice no agrupado en SalesOrderDetail

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderDetail
        ON Sales.SalesOrderDetail (SalesOrderID) include (ProductID,OrderQty,
UnitPrice);
```



Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

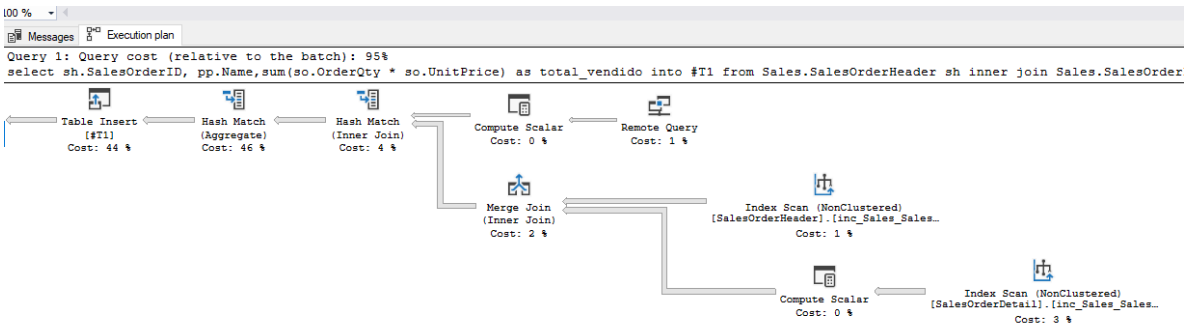
Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.447101 (3%)
Estimated I/O Cost	0.313495
Estimated Subtree Cost	0.447101
Estimated CPU Cost	0.133606
Estimated Number of Executions	1
Estimated Number of Rows to be Read	121317
Estimated Number of Rows for All Executions	121317
Estimated Number of Rows Per Execution	121317
Estimated Row Size	25 B
Ordered	True
Node ID	8

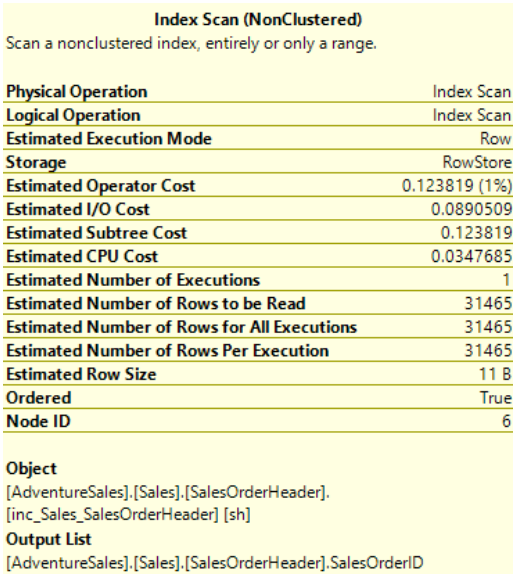
Object
[AdventureSales].[Sales].[SalesOrderDetail].
[inc_Sales_SalesOrderDetail] [so]

Output List
[AdventureSales].[Sales].[SalesOrderDetail].SalesOrderID,
[AdventureSales].[Sales].[SalesOrderDetail].OrderQty,
[AdventureSales].[Sales].[SalesOrderDetail].ProductID,
[AdventureSales].[Sales].[SalesOrderDetail].UnitPrice

Indice no agrupado en SalesOrderHeader

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderHeader
ON Sales.SalesOrderHeader(SalesOrderID) include (TerritoryID,DueDate,TaxAmt);
```





Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.123819 (1%)
Estimated I/O Cost	0.0890509
Estimated Subtree Cost	0.123819
Estimated CPU Cost	0.0347685
Estimated Number of Executions	1
Estimated Number of Rows to be Read	31465
Estimated Number of Rows for All Executions	31465
Estimated Number of Rows Per Execution	31465
Estimated Row Size	11 B
Ordered	True
Node ID	6

Object
[AdventureSales].[Sales].[SalesOrderHeader].
[inc_Sales_SalesOrderHeader] [sh]
Output List
[AdventureSales].[Sales].[SalesOrderHeader].SalesOrderID

En esta consulta se observó que los índices no agrupados en la tabla SalesOrderHeader y SlesOrderDetail redujeron el costo estimado que presentaban los índices agrupados en las llaves primarias, el cual paso de en la tabla Header de 3% a 1% y en la tabla Detail de 6% a 2% manteniendo la secuencia y operadores en ambas consultas.

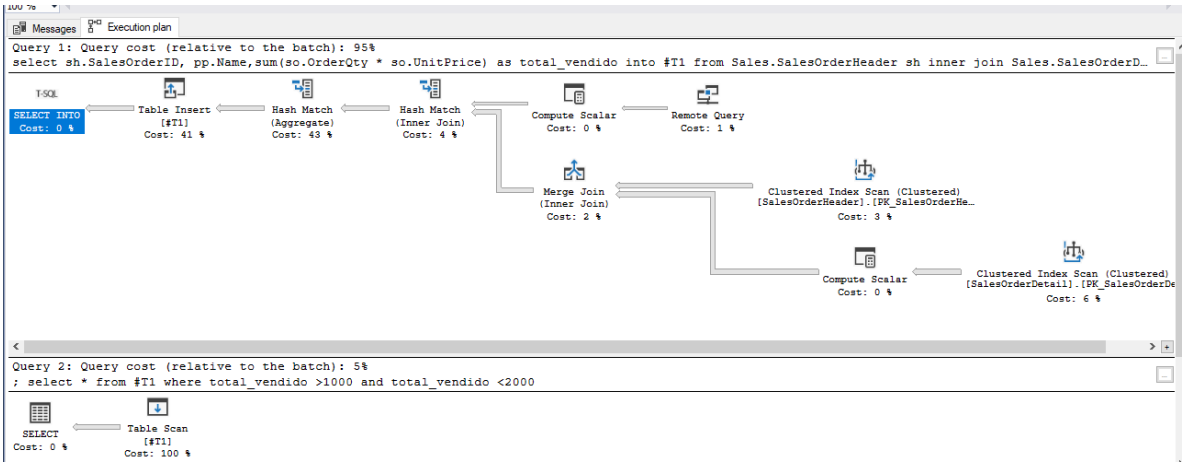
La consulta 6 y 7 es similar a la consulta 5, el único cambio es el intervalo que se pide durante en la compra por lo que se esperan resultados iguales o muy cercanos a esta consulta ya optimizada, por lo que se usaran los mismos índices en las tablas SalesOrderHeader y SlesOrderDetail.

Consulta 6

6. Determinar el número de ventas donde el total de la compra sea mayor a 1000 y menor a 2000.

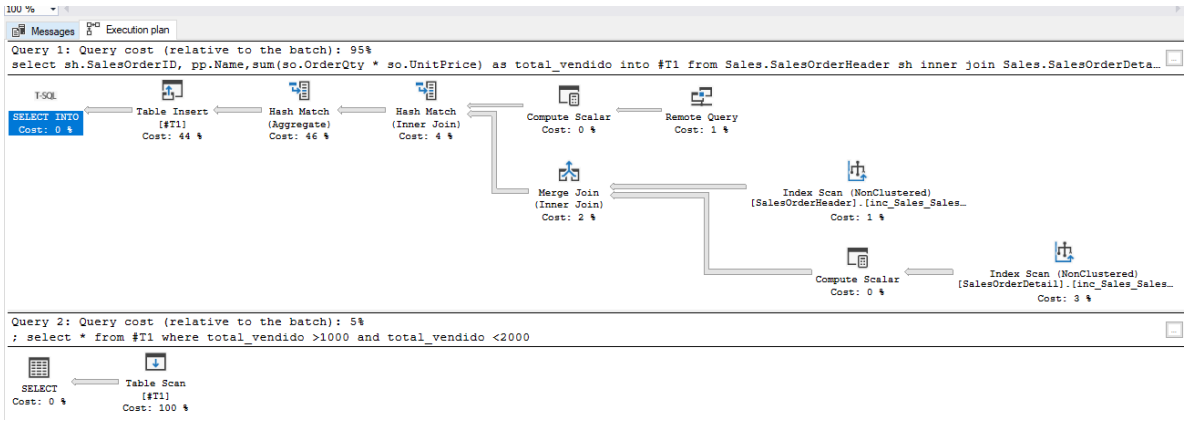
```
select sh.SalesOrderID,  pp.Name,sum(so.OrderQty * so.UnitPrice)
astotal_vendido into #T1
from SQLS2.AdventureSales.Sales.SalesOrderHeader sh
inner join SQLS2.AdventureSales.Sales.SalesOrderDetail so
on sh.SalesOrderID = so.SalesOrderID
inner join Production.Product pp
on pp.ProductID = so.ProductID
group by sh.SalesOrderID, pp.Name
order by sum(so.OrderQty * so.UnitPrice) asc;

select * from #T1
where total_vendido >1000 and total_vendido <2000;
```

Índices agrupados en las tablas SalesOrderDetail y SalesOrderHeader por medio de las llaves primarias:



Índices no agrupados en las tablas SalesOrderDetail y SalesOrderHeader:

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderDetail
ON Sales.SalesOrderDetail (SalesOrderID) include (ProductID,OrderQty,
UnitPrice);

create NONCLUSTERED INDEX inc_Sales_SalesOrderHeader
ON Sales.SalesOrderHeader(SalesOrderID) include (TerritoryID,DueDate,TaxAmt);
```



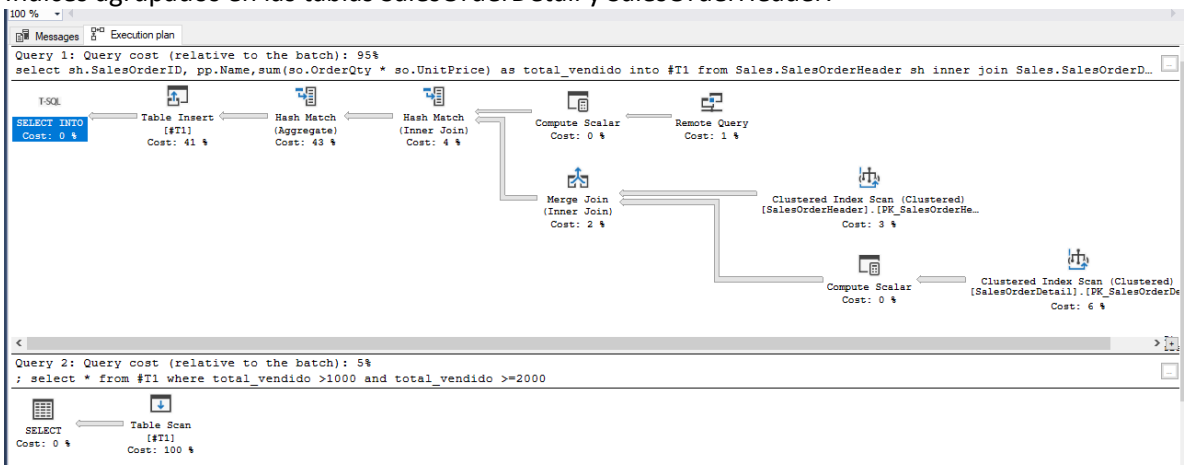
Resultados de optimización con operadores y porcentajes de costo iguales a la consulta 5.

Consulta 7

7. Determinar el número de ventas donde el total de la compra sea mayor a 2000.

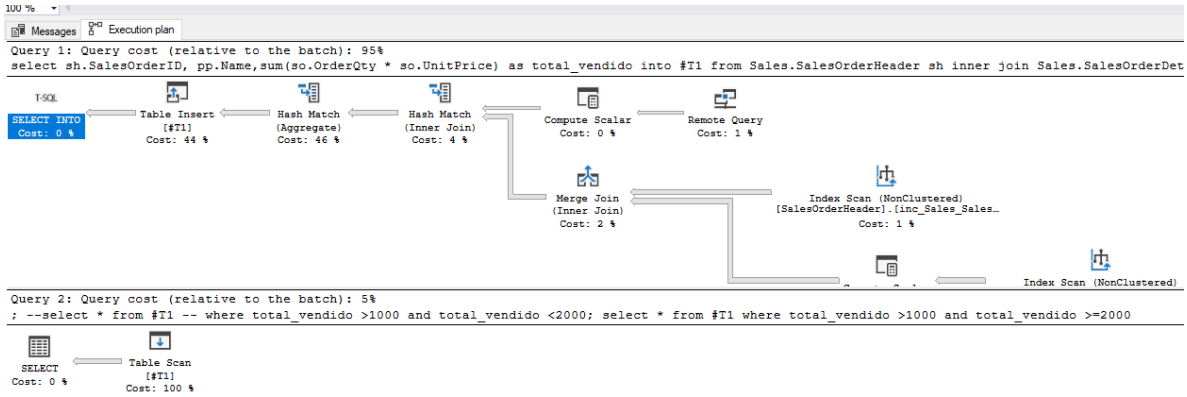
```
select sh.SalesOrderID,  pp.Name,sum(so.OrderQty * so.UnitPrice) as
total_vendido into #T1
from SQLS2.AdventureSales.Sales.SalesOrderHeader sh
inner join SQLS2.AdventureSales.Sales.SalesOrderDetail so
on sh.SalesOrderID = so.SalesOrderID
inner join Production.Product pp
on pp.ProductID = so.ProductID
group by sh.SalesOrderID, pp.Name
order by sum(so.OrderQty * so.UnitPrice) asc;

select * from #T1
where total_vendido >1000 and total_vendido >=2000;
```

Índices agrupados en las tablas SalesOrderDetail y SalesOrderHeader:



Índices no agrupados en las tablas SalesOrderDetail y SalesOrderHeader:

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderDetail
ON Sales.SalesOrderDetail (SalesOrderID) include (ProductID,OrderQty,
UnitPrice);

create NONCLUSTERED INDEX inc_Sales_SalesOrderHeader
ON Sales.SalesOrderHeader(SalesOrderID) include (TerritoryID,DueDate,TaxAmt);
```

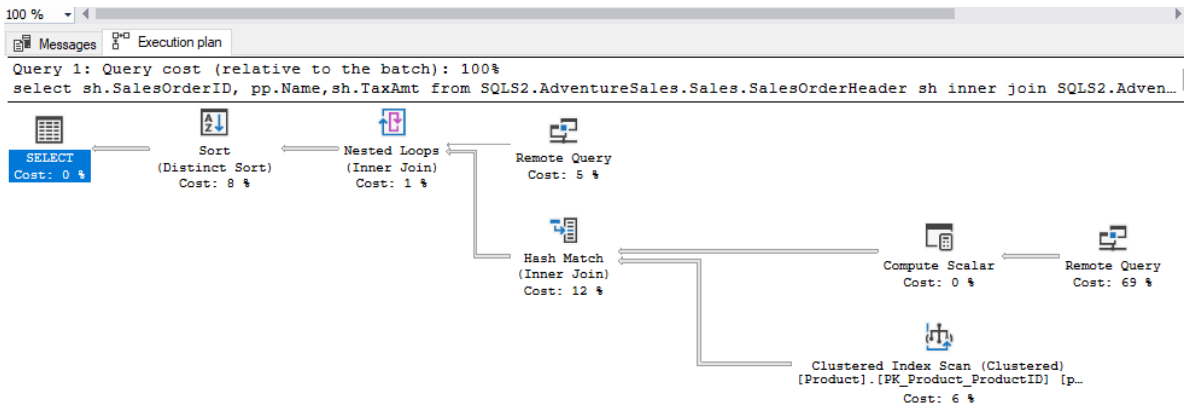


Resultados de optimización con operadores y porcentajes de costo iguales a las consultas 5 y 6.

Consulta 8

8. Determinar las ventas donde los impuestos aplicados sean mayor o igual a los 500 y el territorio sea el 3.

```
select sh.SalesOrderID,  pp.Name,sh.TaxAmt
from SQLS2.AdventureSales.Sales.SalesOrderHeader sh
inner join SQLS2.AdventureSales.Sales.SalesOrderDetail so
on sh.SalesOrderID = so.SalesOrderID
inner join Production.Product pp
on pp.ProductID = so.ProductID
inner join SQLS2.AdventureSales.Sales.SalesTerritory st
on st.TerritoryID = sh.TerritoryID
where TaxAmt >=500 and st.TerritoryID = 3
group by sh.SalesOrderID, pp.Name,TaxAmt
```



Clustered Index Scan (Clustered)

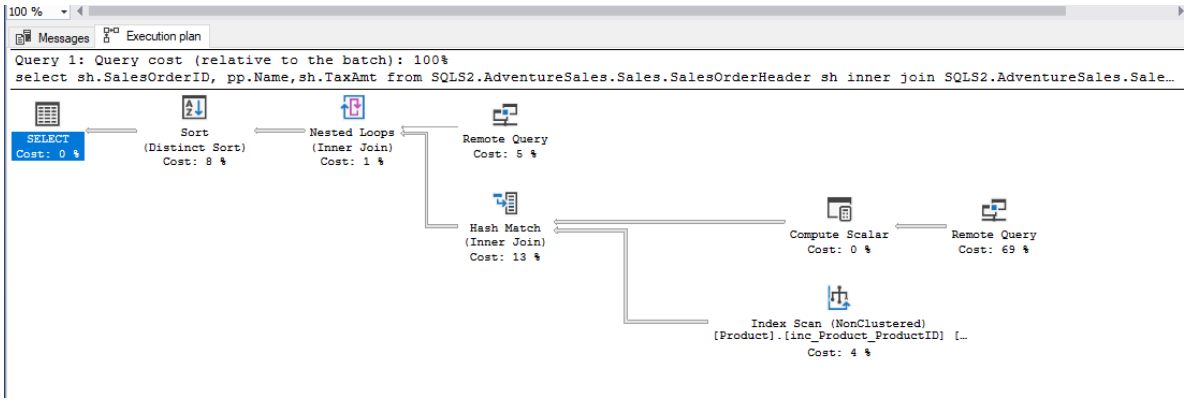
Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated I/O Cost	0.0120139
Estimated Operator Cost	0.0127253 (6%)
Estimated CPU Cost	0.0007114
Estimated Subtree Cost	0.0127253
Estimated Number of Executions	1
Estimated Number of Rows for All Executions	504
Estimated Number of Rows Per Execution	504
Estimated Number of Rows to be Read	504
Estimated Row Size	51 B
Ordered	False
Node ID	6

Object
[Adventure].[Production].[Product].[PK_Product_ProductID] [pp]
Output List
[Adventure].[Production].[Product].ProductID, [Adventure].[Production].[Product].Name

Índice no agrupado en la tabla Product

```
create NONCLUSTERED INDEX inc_Product_ProductID
    ON Production.Product (ProductID)
       include (Name , ProductNumber, ListPrice, Color,
SafetyStockLevel,StandardCost,DaysToManufacture);
GO
```



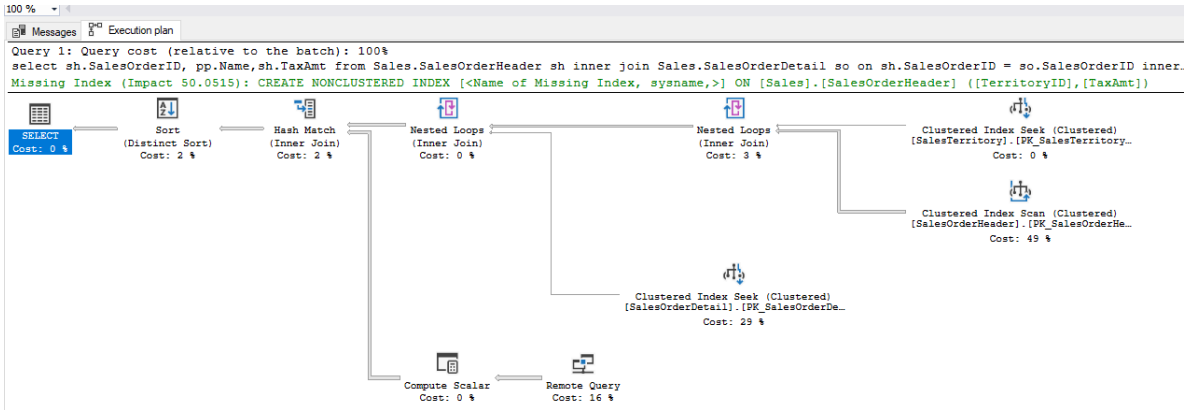
Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0082808 (4%)
Estimated I/O Cost	0.0075694
Estimated Subtree Cost	0.0082808
Estimated CPU Cost	0.0007114
Estimated Number of Executions	1
Estimated Number of Rows to be Read	504
Estimated Number of Rows for All Executions	504
Estimated Number of Rows Per Execution	504
Estimated Row Size	51 B
Ordered	False
Node ID	6

Object
[Adventure].[Production].[Product].[inc_Product_ProductID] [pp]
Output List
[Adventure].[Production].[Product].ProductID, [Adventure].
[Production].[Product].Name

Consulta ejecutada desde la segunda instancia:
Se muestran índices agrupados por llave primaria en las tablas SalesTerritory, SalesSalesOrderHeader y SalesOrderDetail.



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.542338 (49%)
Estimated I/O Cost	0.507569
Estimated Subtree Cost	0.542338
Estimated CPU Cost	0.0347685
Estimated Number of Executions	1
Estimated Number of Rows to be Read	31465
Estimated Number of Rows for All Executions	101.673
Estimated Number of Rows Per Execution	101.673
Estimated Row Size	23 B
Ordered	False
Node ID	6

Predicate
[AdventureSales].[Sales].[SalesOrderHeader].[TerritoryID] as [sh].
[TerritoryID]=(3) AND [AdventureSales].[Sales].[SalesOrderHeader].
[TaxAmt] as [sh].[TaxAmt]>=($500.0000)
Object
[AdventureSales].[Sales].[SalesOrderHeader].
[PK_SalesOrderHeader_SalesOrderID] [sh]
Output List
[AdventureSales].[Sales].[SalesOrderHeader].SalesOrderID,
[AdventureSales].[Sales].[SalesOrderHeader].TaxAmt

Clustered Index Seek (Clustered)

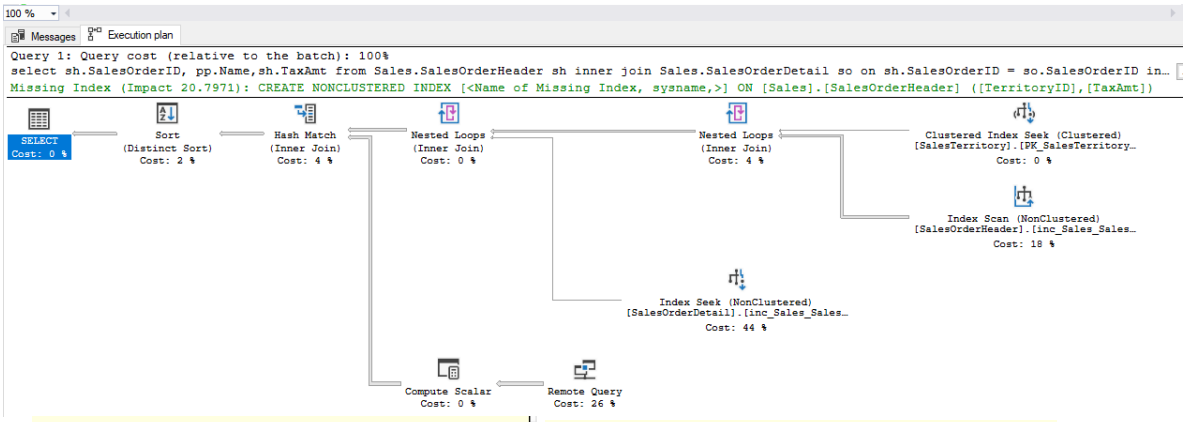
Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.318653 (29%)
Estimated I/O Cost	0.003125
Estimated Subtree Cost	0.318653
Estimated CPU Cost	0.0001616
Estimated Number of Executions	101.673
Estimated Number of Rows to be Read	4.21226
Estimated Number of Rows for All Executions	428.273111
Estimated Number of Rows Per Execution	4.21226
Estimated Row Size	11 B
Ordered	True
Node ID	7

Object
[AdventureSales].[Sales].[SalesOrderDetail].
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] [so]
Output List
[AdventureSales].[Sales].[SalesOrderDetail].ProductID
Seek Predicates
Seek Keys[1]: Prefix: [AdventureSales].[Sales].
[SalesOrderDetail].SalesOrderID = Scalar Operator([AdventureSales].
[Sales].[SalesOrderHeader].[SalesOrderID] as [sh].[SalesOrderID])

Se crean dos índices no agrupados para eficientar la consulta (mismos en las consultas anteriores) en las tablas SalesSalesOrderHeader y SalesOrderDetail.

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderDetail
ON Sales.SalesOrderDetail (SalesOrderID) include (ProductID,OrderQty,
UnitPrice);

create NONCLUSTERED INDEX inc_Sales_SalesOrderHeader
ON Sales.SalesOrderHeader(SalesOrderID) include (TerritoryID,DueDate,TaxAmt);
```



Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.123819 (18%)
Estimated I/O Cost	0.0890509
Estimated Subtree Cost	0.123819
Estimated CPU Cost	0.0347685
Estimated Number of Executions	1
Estimated Number of Rows to be Read	31465
Estimated Number of Rows for All Executions	101.673
Estimated Number of Rows Per Execution	101.673
Estimated Row Size	23 B
Ordered	False
Node ID	6

Predicate
[AdventureSales].[Sales].[SalesOrderHeader].[TerritoryID] as [sh].[TerritoryID]=(3) AND [AdventureSales].[Sales].[SalesOrderHeader].[TaxAmt] as [sh].[TaxAmt]>=($500.0000)
Object
[AdventureSales].[Sales].[SalesOrderHeader].[inc_Sales_SalesOrderHeader] [sh]
Output List
[AdventureSales].[Sales].[SalesOrderHeader].SalesOrderID, [AdventureSales].[Sales].[SalesOrderHeader].TaxAmt

Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.296427 (44%)
Estimated I/O Cost	0.003125
Estimated Subtree Cost	0.296427
Estimated CPU Cost	0.0001612
Estimated Number of Executions	101.673
Estimated Number of Rows to be Read	3.85562
Estimated Number of Rows for All Executions	392.0124523
Estimated Number of Rows Per Execution	3.85562
Estimated Row Size	11 B
Ordered	True
Node ID	7

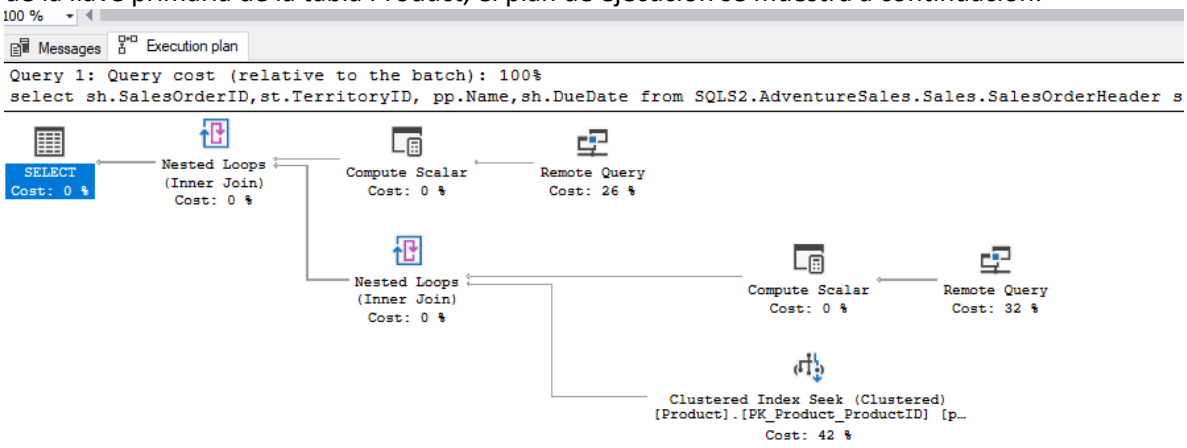
Object
[AdventureSales].[Sales].[SalesOrderDetail].[inc_Sales_SalesOrderDetail] [so]
Output List
[AdventureSales].[Sales].[SalesOrderDetail].ProductID
Seek Predicates
Seek Keys[1]: Prefix: [AdventureSales].[Sales].[SalesOrderDetail].SalesOrderID = Scalar Operator([AdventureSales].[Sales].[SalesOrderHeader].[SalesOrderID] as [sh].[SalesOrderID])

Consulta 9

9. Listar las órdenes del territorio 9 con fecha de modificación del 2011-06-30 00:00:00.000

```
select sh.SalesOrderID,st.TerritoryID,  pp.Name,sh.DueDate
from SQLS2.AdventureSales.Sales.SalesOrderHeader sh
inner join SQLS2.AdventureSales.Sales.SalesOrderDetail so
on sh.SalesOrderID = so.SalesOrderID
inner join Production.Product pp
on pp.ProductID = so.ProductID
inner join SQLS2.AdventureSales.Sales.SalesTerritory st
on st.TerritoryID = sh.TerritoryID
where DueDate ='2011-06-30 00:00:00.000' and st.TerritoryID = 9;
```

Ejecución de la consulta desde el servidor 1, el cual contiene un índice agrupado por medio de la llave primaria de la tabla Product, el plan de ejecución se muestra a continuación:



Clustered Index Seek (Clustered)

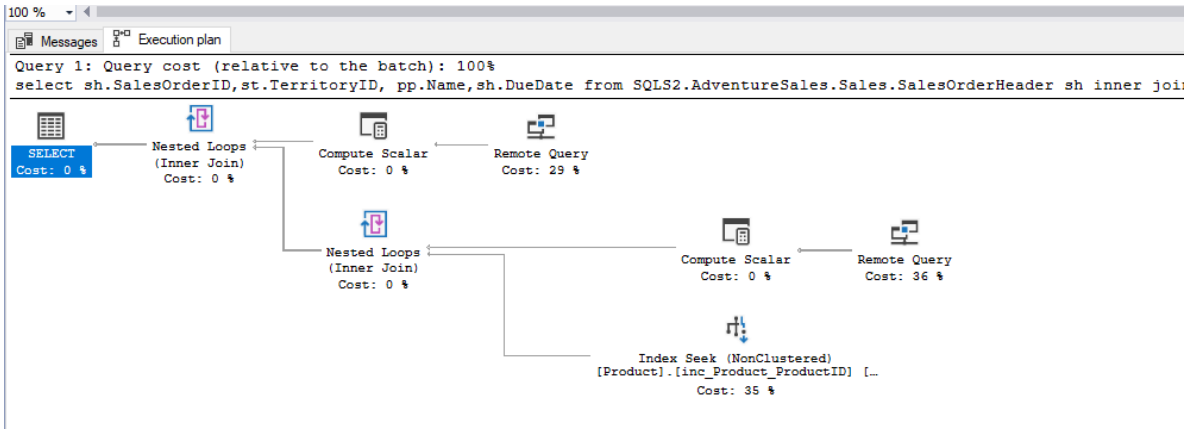
Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0170278 (42%)
Estimated I/O Cost	0.003125
Estimated Subtree Cost	0.0170278
Estimated CPU Cost	0.0001581
Estimated Number of Executions	8.6993796
Estimated Number of Rows to be Read	1
Estimated Number of Rows for All Executions	8.6993796
Estimated Number of Rows Per Execution	1
Estimated Row Size	61 B
Ordered	True
Node ID	6

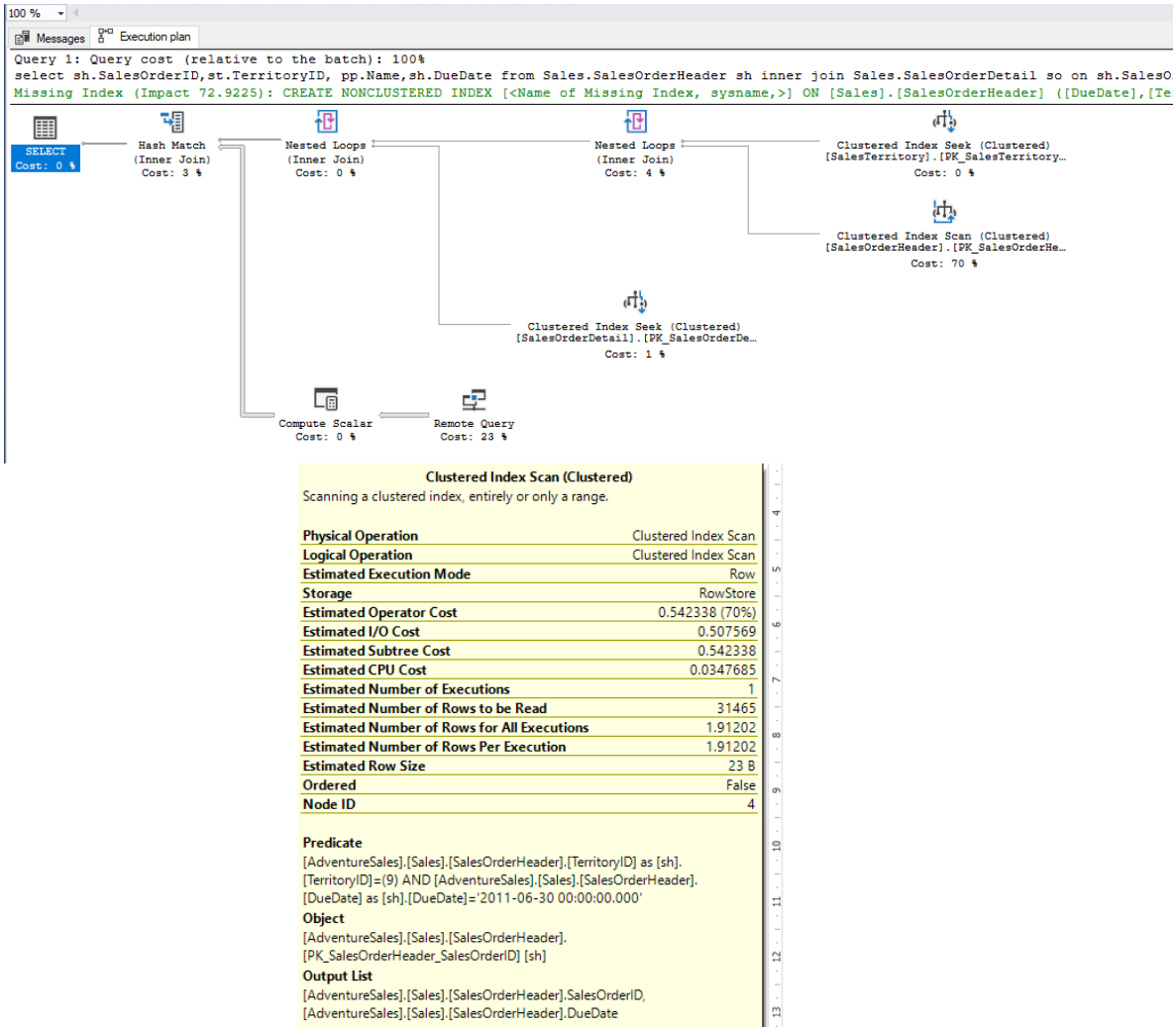
Object
[Adventure].[Production].[Product].[PK_Product_ProductID] [pp]
Output List
[Adventure].[Production].[Product].Name
Seek Predicates
Seek Keys[1]: Prefix: [Adventure].[Production].[Product].ProductID = Scalar Operator([SQLS2].[AdventureSales].[Sales].[SalesOrderDetail].[ProductID] as [so].[ProductID])

Incluyendo el índice no agrupado en la tabla Prodcut se obtiene el siguiente plan de ejecución:

```
create NONCLUSTERED INDEX inc_Product_ProductID
ON Production.Product (ProductID)
include (Name , ProductNumber, ListPrice, Color,
SafetyStockLevel,StandardCost,DaysToManufacture);
```



Analizando la consulta distribuida desde el segundo servidor tenemos el siguiente plan de ejecución tomando en cuenta los índices agrupados por llave primaria de las tablas SlesTerritory, SalesOrderHeader y SalesOrderDetail:



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.542338 (70%)
Estimated I/O Cost	0.507569
Estimated Subtree Cost	0.542338
Estimated CPU Cost	0.0347685
Estimated Number of Executions	1
Estimated Number of Rows to be Read	31465
Estimated Number of Rows for All Executions	1.91202
Estimated Number of Rows Per Execution	1.91202
Estimated Row Size	23 B
Ordered	False
Node ID	4

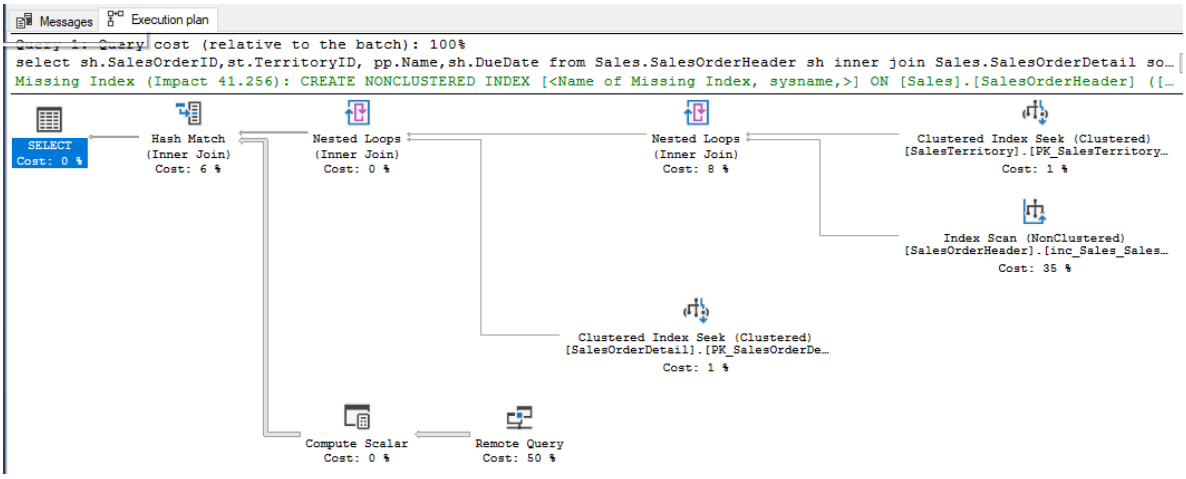
Predicate
[AdventureSales].[Sales].[SalesOrderHeader].[TerritoryID] as [sh].[TerritoryID]=(9) AND [AdventureSales].[Sales].[SalesOrderHeader].[DueDate] as [sh].[DueDate]='2011-06-30 00:00:00.000'

Object
[AdventureSales].[Sales].[SalesOrderHeader].[PK_SalesOrderHeader_SalesOrderID] [sh]

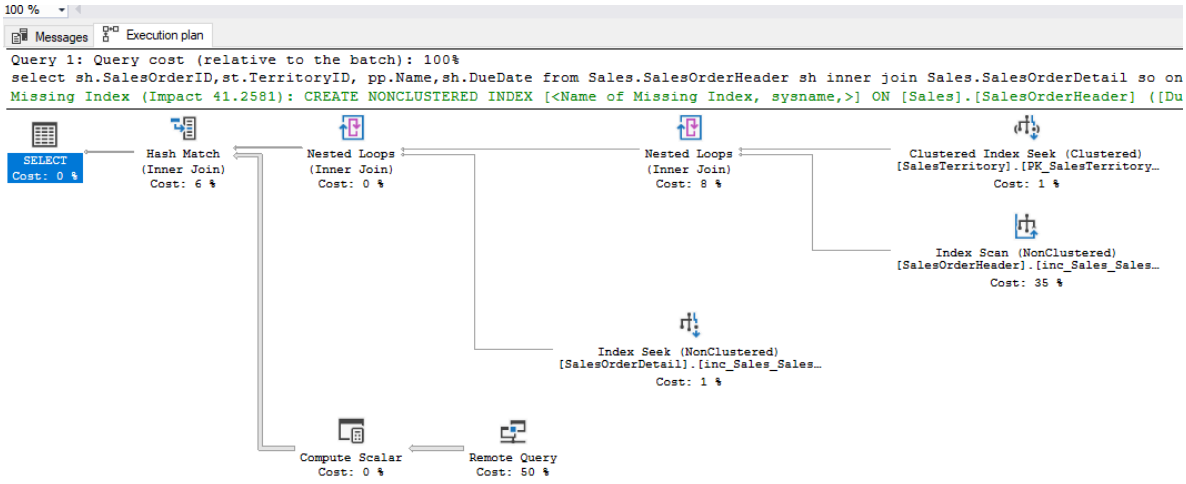
Output List
[AdventureSales].[Sales].[SalesOrderHeader].SalesOrderID, [AdventureSales].[Sales].[SalesOrderHeader].DueDate

Creando un índice no agrupado en la tabla SalesOrderHeader, se obtiene el siguiente plan de ejecución donde se observa que el coste ha disminuido en la tabla SalesOrederHeader:

```
create NONCLUSTERED INDEX inc_Sales_SalesOrderHeader
ON Sales.SalesOrderHeader(SalesOrderID) include (TerritoryID,DueDate,TaxAmt);
```



En caso de implementar un segundo índice agrupado en la tabla SalesOrderDetail, mantendría los operadores y el mismo coste:



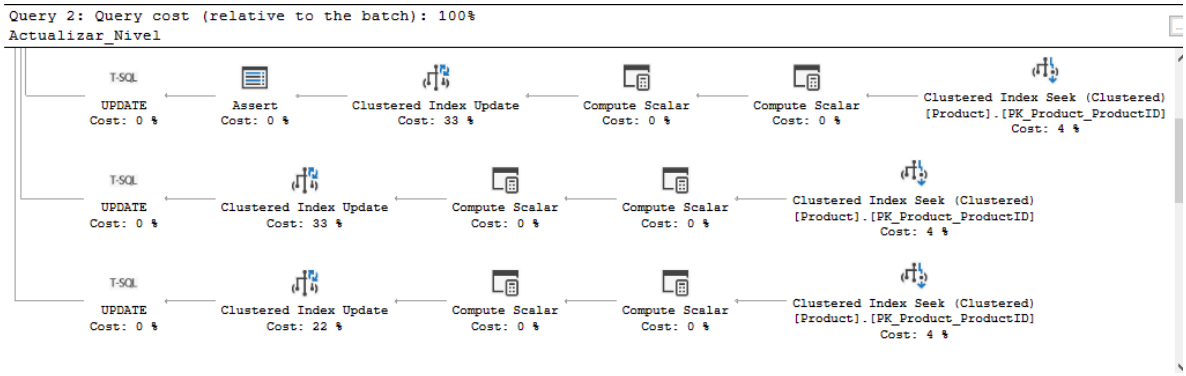
Consulta 10

10. Actualizar el nivel de existencias de seguridad y el color (SafetyStockLevel) del producto indicado, así como la fecha de modificación.

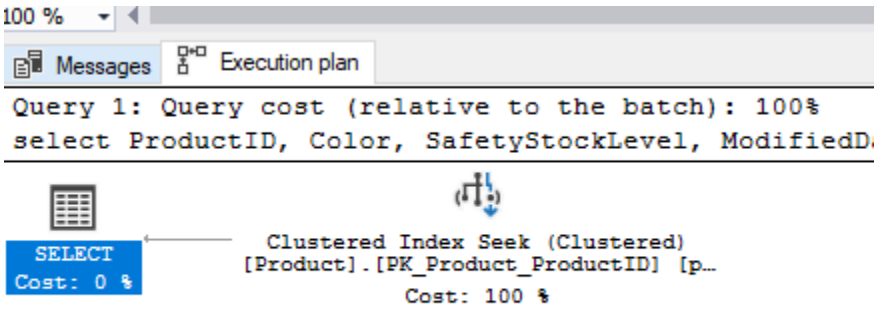
```
alter procedure Actualizar_Nivel

                @ID varchar(5),
                @Nivel smallint,
                @Col nvarchar(15)

                as
                begin TRANSACTION
                BEGIN TRY
                UPDATE Production.Product set SafetyStockLevel = @Nivel
where ProductID = @ID;
                UPDATE Production.Product set Color = @Col where
ProductID= @ID;
                UPDATE Production.Product set ModifiedDate = getdate()
where ProductID= @ID;
                commit transaction
                end try
```

```
        begin catch
        rollback transaction
        print 'Ocurrio un error'
        end catch
go
exec Actualizar_Nivel '707', '4','Yellow'

select ProductID, Color, SafetyStockLevel, ModifiedDate
from Production.Product pp where ProductID = '707';
```

Para esta transacción únicamente se utilizó un índice agrupado en la tabla Product ya que la búsqueda se realiza por medio del ID para modificar los campos solicitados.




```
select ProductID, Color, SafetyStockLevel, ModifiedDate
                    from Production.Product pp where ProductID = '707';
```



En esta consulta usa el índice agrupado por ProductID

Consulta 11

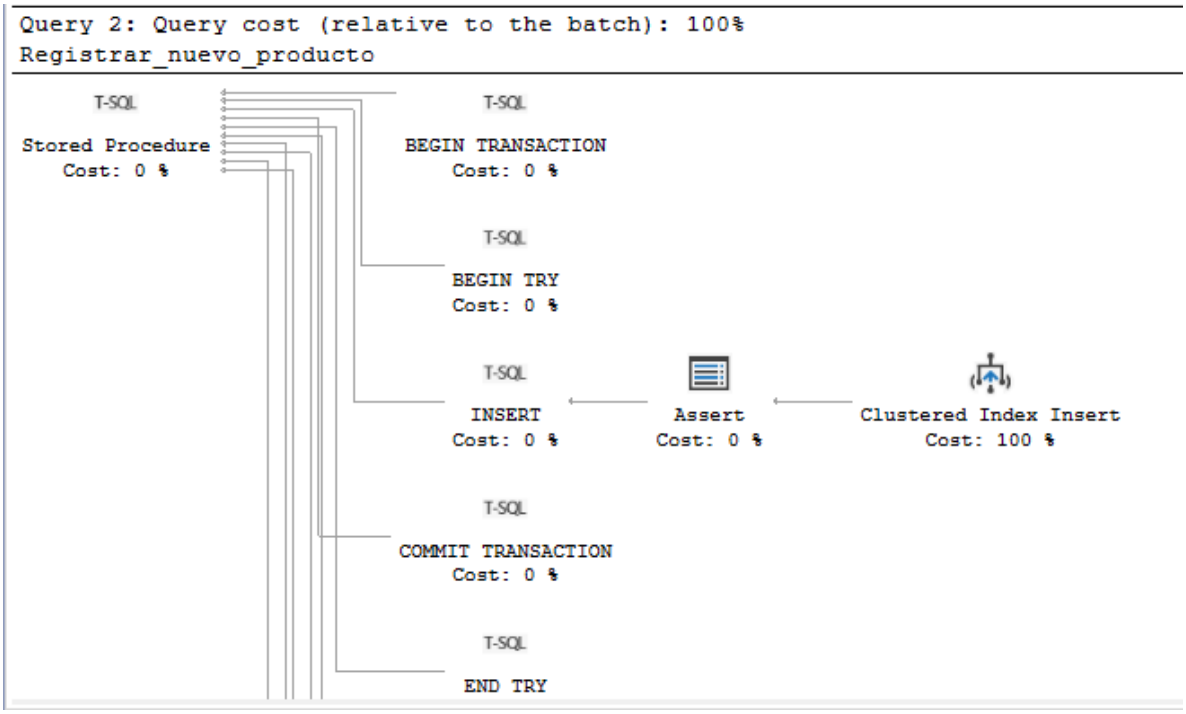
11. Registrar un nuevo producto.

```
SET IDENTITY_INSERT Production.Product ON ;
      create procedure Registrar_nuevo_producto

                                @ProductIDx int,
                                @Namex nvarchar(50),
                                @ProductNumberx nvarchar(25),
                                @MakeFlagx bit,
                                @FinishedGoodsFlagx bit,
                                @Colorx nvarchar(50),
                                @SafetyStockLevelx smallint,
```

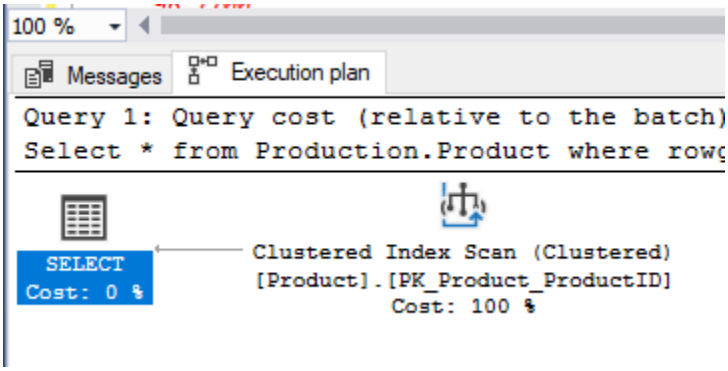
```
                    @ReorderPointx smallint,
                    @StandardCostx money,
                    @ListPricex money,
                    @Sizex nvarchar(5),
                    @SizeUnitMeasureCodex nchar(3),
                    @WeightUnitMeasureCodex nchar(3),
                    @Weightx decimal(8,2),
                    @DaysToManufacturex int,
                    @ProductLinex nchar(2),
                    @Classx nchar(2),
                    @Stylex nchar(2),
                    @ProductSubcategoryIDx int,
                    @ProductModelIDx int,
                    @rowguidx uniqueidentifier

                  as
                  begin TRANSACTION
                  BEGIN TRY
                  insert into Production.Product(
                     [ProductID]
                    ,[Name]
                    ,[ProductNumber]
                    ,[MakeFlag]
                    ,[FinishedGoodsFlag]
                    ,[Color]
                    ,[SafetyStockLevel]
                    ,[ReorderPoint]
                    ,[StandardCost]
                    ,[ListPrice]
                    ,[Size]
                    ,[SizeUnitMeasureCode]
                    ,[WeightUnitMeasureCode]
                    ,[Weight]
                    ,[DaysToManufacture]
                    ,[ProductLine]
                    ,[Class]
                    ,[Style]
                    ,[ProductSubcategoryID]
                    ,[ProductModelID]
                    ,[SellStartDate]
                    ,[SellEndDate]
                    ,[DiscontinuedDate]
                    ,[rowguid]
                    ,[ModifiedDate])


            Values (
                     @ProductIDx,
                          @Namex ,
                          @ProductNumberx ,
                          @MakeFlagx ,
                          @FinishedGoodsFlagx,
                          @Colorx ,
                          @SafetyStockLevelx,
                          @ReorderPointx ,
                          @StandardCostx ,
                          @ListPricex ,
                          @Sizex ,
                          @SizeUnitMeasureCodex ,
                          @WeightUnitMeasureCodex ,
                          @Weightx ,
```

```
                                @DaysToManufacturex ,
                                @ProductLinex ,
                                @Classx ,
                                @Stylex ,
                                @ProductSubcategoryIDx ,
                                @ProductModelIDx ,
                                GETDATE(),
                                NULL ,
                                NULL ,
                                @rowguidx ,
                                 GETDATE());
                            commit transaction
                            end try

                            begin catch
                            rollback transaction
                            print 'Ocurrio un error'
                            select ERROR_MESSAGE()
                            end catch
                        go

exec Registrar_nuevo_producto
       '1000',
       'Nuevo2',
       'AR-5382',
       '0',
       '0',
       'black',
       '1000',
       '600',
       '98.7700',
       '147.1400',
       'G',
       'G',
       'G',
       '445.00',
       '1',
       'M',
       'L',
       'U',
       '1',
       '23',
       '75752E26-A3B6-4264-9B06-F23A4FBDC5A0'

Select * from Production.Product where rowguid='75752E26-A3B6-4264-9B06-
F23A4FBDC5A0'
```

Al hacer uso del Id del producto usa un índice agrupado para realizar la inserción de un nuevo producto:




```
Select * from Production.Product where rowguid='75752E26-A3B6-4264-9B06-
F23A4FBDC5A0'
```

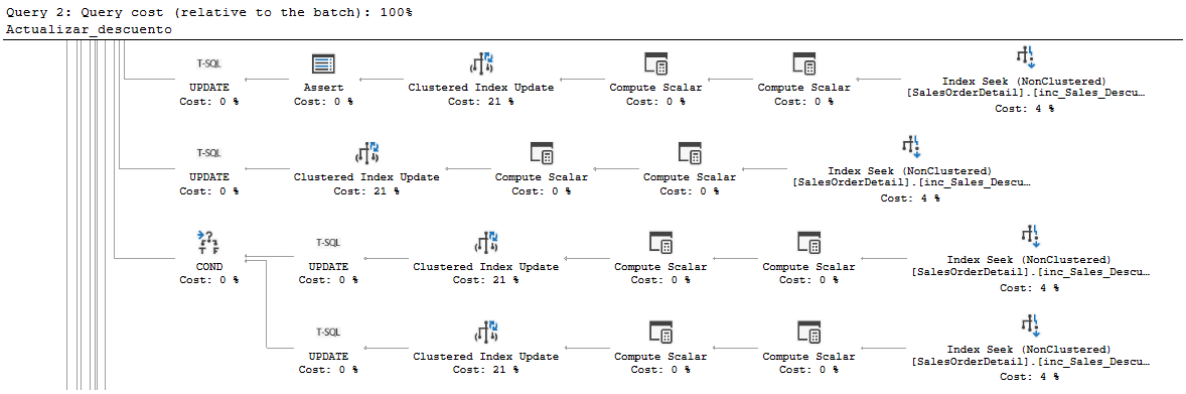


Consulta 12

12. Actualizar el descuento del detalle de la orden de compra y el ID de oferta.

```
                    alter procedure Actualizar_descuento

                          @SalesOrderDetailIDx tinyint,
                          @UnitPriceDiscountx money

                    as
                          begin TRANSACTION
                          BEGIN TRY
                          UPDATE Sales.SalesOrderDetail set UnitPriceDiscount =
@UnitPriceDiscountx  where SalesOrderDetailID = @SalesOrderDetailIDx;
                          UPDATE Sales.SalesOrderDetail set ModifiedDate =
getdate() where SalesOrderDetailID = @SalesOrderDetailIDx;
                          if(@UnitPriceDiscountx = 0.00)
                          UPDATE Sales.SalesOrderDetail set SpecialOfferID = '1'
where SalesOrderDetailID = @SalesOrderDetailIDx;
```

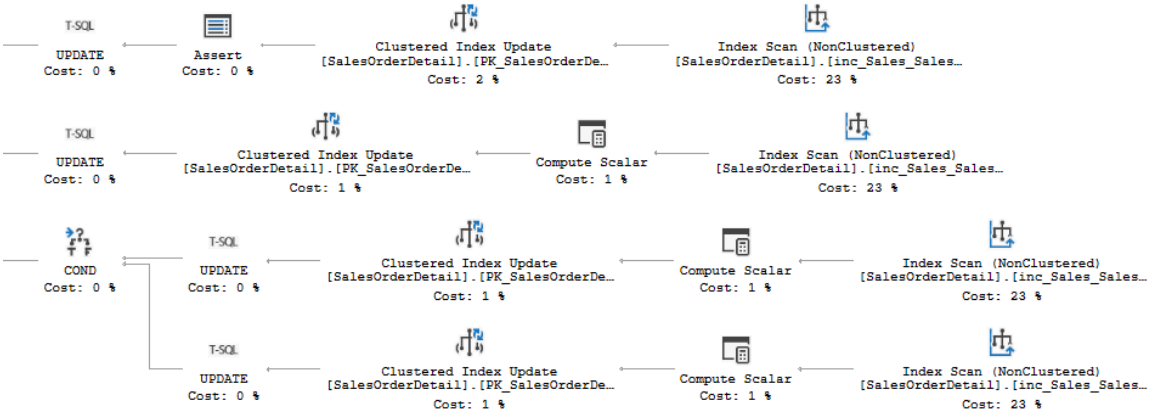
```
                        else
                        UPDATE Sales.SalesOrderDetail set SpecialOfferID = '17'
where SalesOrderDetailID = @SalesOrderDetailIDx;
                        commit transaction
                        end try
                        begin catch
                        rollback transaction
                        print 'Ocurrio un error'
                        select ERROR_MESSAGE()

                        end catch
                go

                exec Actualizar_descuento '1', '0.00'

            select * from Sales.SalesOrderDetail where SalesOrderDetailID = 1;
```



Esta inserción se logra eficientar creado un índice no agrupado en la tabla SalesOrderDetail donde se incluyan los campos a utilizar en dicha operación.

Agregando índice no agrupado

```
create NONCLUSTERED INDEX inc_Sales_Descuenyo
ON Sales.SalesOrderDetail (SalesOrderDetailID) include
(UnitPriceDiscount,SpecialOfferID, ModifiedDate);
```



```
select * from Sales.SalesOrderDetail where SalesOrderDetailID = 1;
```

100 %

Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
select * from Sales.SalesOrderDetail where SalesOrderDetailID = 1
```