UCR - ECCI - Bases de Datos
Fabián Orozco Chaves - B95690
Daniel Escobar Giraldo - C02748
Grupo 01 - II Ciclo 2022
Laboratorio 7 - Optimización de consultas.

Ejercicio 1. a). Ejecute las siguientes consultas (marque ambas y ejecútelas juntas):

```
SELECT ProductID FROM Sales.SalesOrderDetail;
SELECT ProductID FROM dbo.SalesOrderDetail;
```

RESULTS

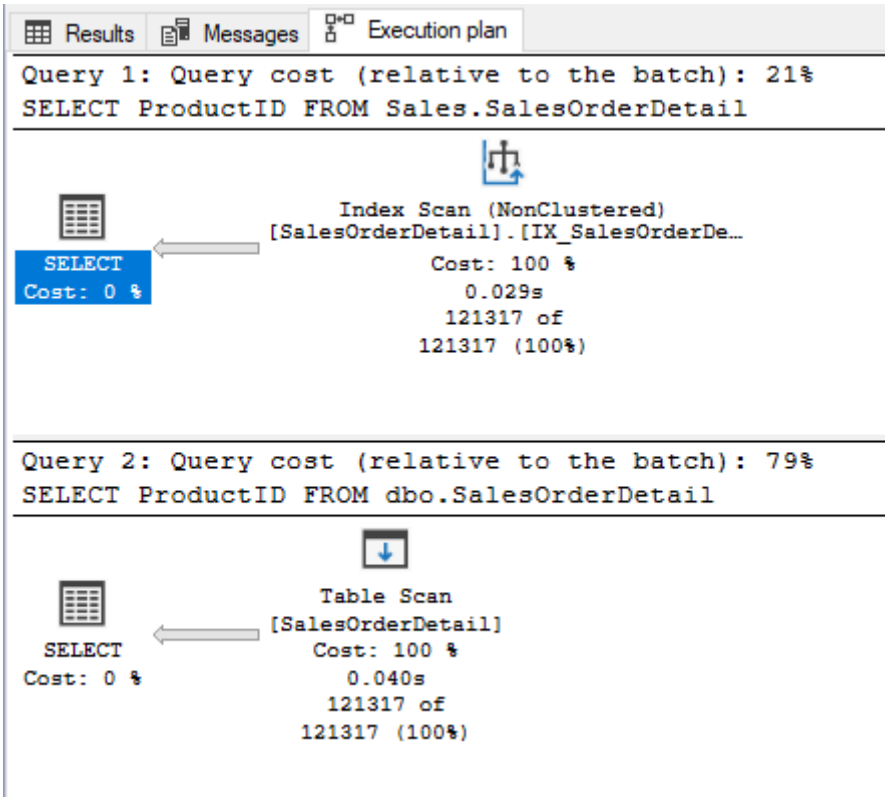
	ProductID
1	707
2	707
3	707
4	707
5	707
6	707
7	707
8	707
9	707
10	707
11	707
12	707

	ProductID
1	715
2	772
3	716
4	729
5	732
6	766
7	767
8	758
9	761
10	751
11	754
12	757

MESSAGES

[9:54:09 AM] Started executing query at Line 8
(121317 rows affected)
(121317 rows affected)
Total execution time: 00:00:00.833

Ejercicio 1. b) Muestre el plan de ejecución real para las consultas del punto 1(a).



Ejercicio 1. c) ¿Cuál es el costo porcentual (relativo al lote o batch) de cada una de las consultas?

El costo relativo al batch de la consulta 1 es del 21%.

El costo relativo al batch de la consulta 2 es del 79%.

Ejercicio 1. d) ¿A qué atribuye la gran diferencia entre el costo de la consulta sobre el esquema Sales y la consulta sobre el esquema dbo? Para contestar esta pregunta, analice el plan de ejecución de cada consulta y revise si las tablas involucradas tienen algún índice que esté siendo usado en alguna de las consultas.

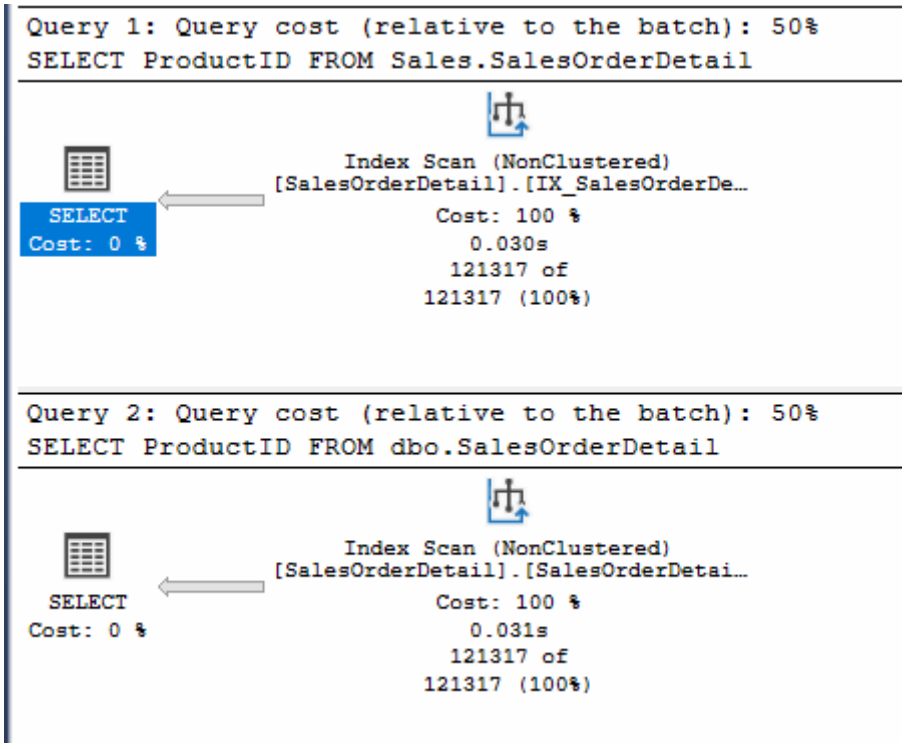
La tabla de la consulta 1 tiene un índice sobre la columna de ProductID por lo que esta consulta es menos costosa ya que lo accede por bloque. Por otro lado la tabla de la consulta 2 no contiene un índice por lo que debe hacer un scan completo de la tabla.

Ejercicio 1. e) Cree un índice sobre la tabla dbo.SalesOrderDetail que permita mejorar el rendimiento de la consulta:

Creación del índice:

```
create index SalesOrderDetail_ProductID
ON dbo.SalesOrderDetail(ProductID)
```

Ejercicio 1. f) Para comprobar si el índice fue efectivo, vuelva a ejecutar las consultas del punto 1(a) y revise si el índice que creó está siendo utilizado en el plan de ejecución. Además, compare el costo porcentual (relativo al lote o batch) de cada consulta y note si hubo algún cambio con lo observado en el punto 1(c). Explique.



Sí hubo un cambio en el costo y con esto se verifica que el índice fue creado. Ahora el costo de ambas consultas es el mismo, por lo que se llevan la mitad del costo cada una de todo el lote.

Ejercicio 1.g) Elimine el índice creado en el punto 1(e).

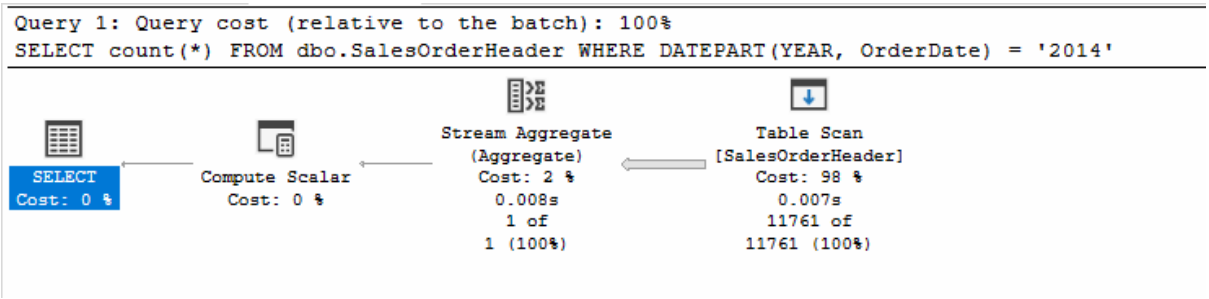
Comando para eliminar el index:

```
drop index SalesOrderDetail_ProductID on dbo.SalesOrderDetail
```

Ejercicio 2. a) Ejecute la siguiente consulta:

```
SELECT count(*)
FROM dbo.SalesOrderHeader
WHERE DATEPART(YEAR, OrderDate) = ‘2014’
```

Ejercicio 2. b). Muestre el plan de ejecución real para la consulta del punto 2(a).



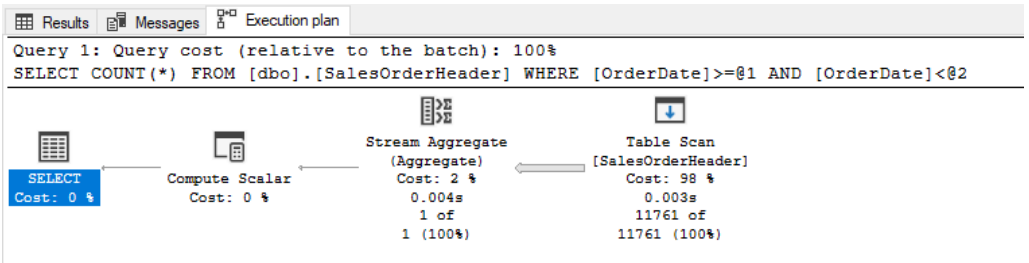
Ejercicio 2. c). ¿Cómo resolvió SQL Server esta consulta? ¿Cuáles operadores físicos usó?

Hizo un escaneo de la tabla completa (utiliza table scan) para recuperar todos los registros de la tabla SalesOrderHeader la cual es filtrada por la función DatePart (utiliza Stream Aggregate), dejando únicamente las fechas con el año 2014; por último, recupera la cantidad (utiliza Compute Scalar) de registros de la tabla resultante.

Ejercicio 2. d) Ahora ejecute esta consulta, que tiene su cláusula WHERE modificada pero da el mismo resultado que la consulta del punto 2(a):

```
SELECT count(*)
FROM dbo.SalesOrderHeader
WHERE OrderDate >= ‘20140101’ AND OrderDate < ‘20150101’;
```

Ejercicio 2. e) Muestre el plan de ejecución real para la consulta del punto 2(d)



Ejercicio 2. f) ¿Cómo resolvió SQL Server esta consulta? ¿Cuáles operadores físicos usó? ¿Qué sugerencia de índice le da SQL Server para optimizar la consulta (i.e., missing index)?

Utiliza los mismos operadores físicos que en la consulta anterior (table scan, Stream Aggregate, Compute Scalar).

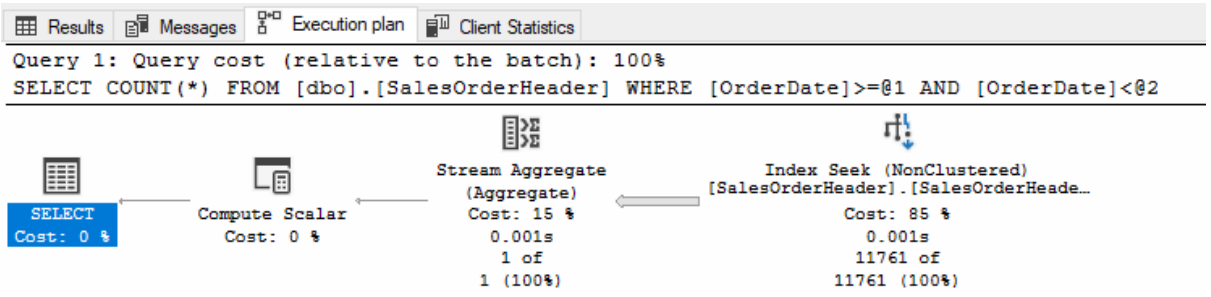
*SQL Server no dio ninguna sugerencia para el índice. (Avalado por la profesora)

Ejercicio 2. g) Cree un índice de acuerdo a la recomendación de SQL Server

*Lo creamos en base a lo que creemos correcto (index sobre OrderDate). (Avalado por la profesora)

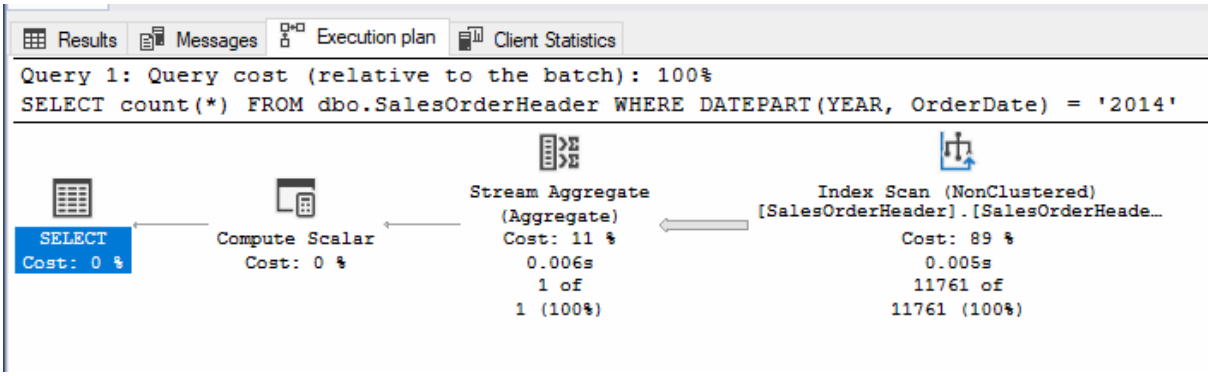
```
create index SalesOrderHeader_OrderDate
on dbo.SalesOrderHeader(OrderDate)
```

Ejercicio 2. h) Vuelva a ejecutar la consulta del punto 2(d) y revise el plan de ejecución. ¿Hay alguna diferencia entre los operadores que está usando este plan de ejecución y el del punto 2(e) –antes de crear el índice–? ¿Está SQL Server usando el índice creado en el punto 2(g)? Explique.



La única diferencia es que se utilizó el operador físico index seek en lugar de table scan. Esto muestra que si se está utilizando el índice.

Ejercicio 2. i) Vuelva a ejecutar la consulta del punto 2(a) y revise el plan de ejecución. ¿Hay alguna diferencia entre los operadores que está usando este plan de ejecución y el del punto 2(b)? ¿Por qué no está SQL Server usando el índice creado en el punto 2(g)? Explique.



Sí hay diferencia respecto a los operadores usados, ya que en el plan de 2(b) utiliza Table Scan y en 2(i) utiliza Index Scan.

SQL Server sí utiliza el índice creado en el punto 2(g) pero cambia el operador (Seek para 2(g) y Scan para 2(i))

seek sí va a la tabla (usa tipo búsqueda binaria)

scan no va a la tabla, es más caro

Ejercicio 2. j) Elimine el índice creado en el punto 2(g).

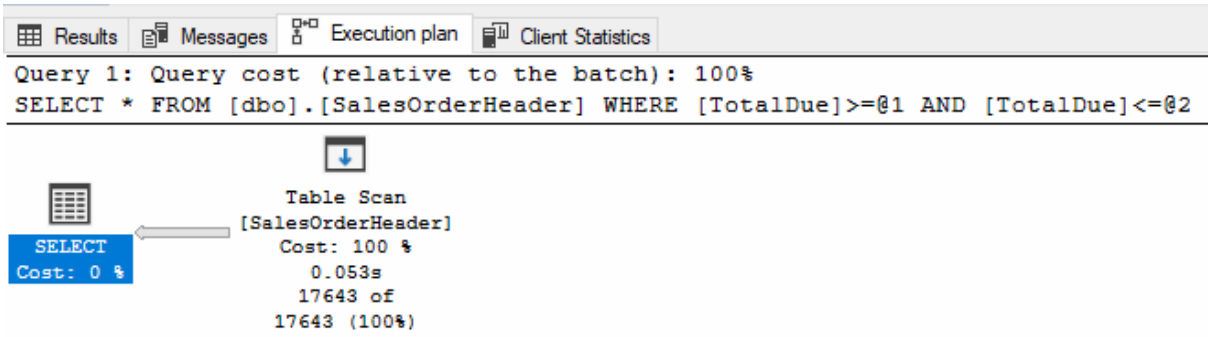
Comando utilizado para eliminar el index:

```
drop index SalesOrderHeader_OrderDate on dbo.SalesOrderHeader
```

Ejercicio 3. a) Ejecute la siguiente consulta:

```
SELECT * FROM dbo.SalesOrderHeader
WHERE TotalDue BETWEEN 500 AND 40000;
```

Ejercicio 3. b) Muestre el plan de ejecución real para la consulta del punto 3(a).



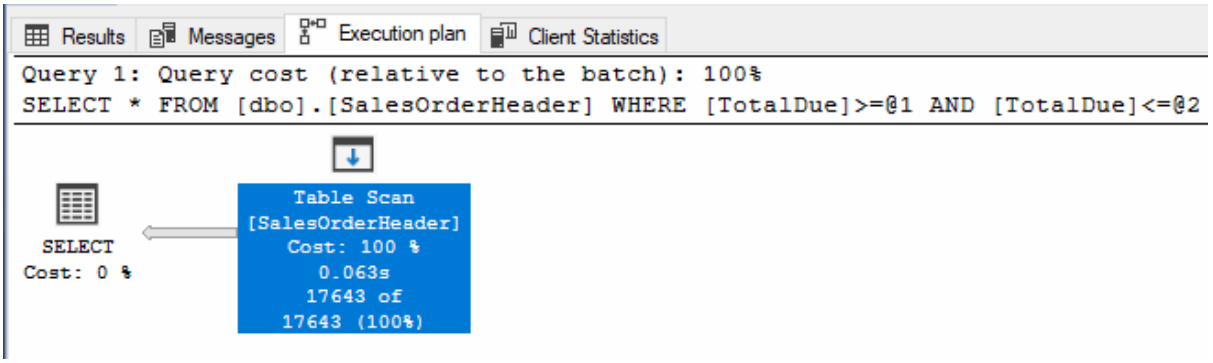
Ejercicio 3. c) ¿Cómo resolvió SQL Server esta consulta? ¿Cuáles operadores físicos usó?

SQL Server resolvió la consulta utilizando el operador físico table scan, reemplazando el operador between por los operadores de comparación >= y <=.

Ejercicio 3. d) Cree un índice sobre el atributo TotalDue de la tabla dbo.SalesOrderHeader para mejorar el rendimiento de la consulta.

```
create index SalesOrderHeader_TotalDue on dbo.SalesOrderHeader(TotalDue)
```

Ejercicio 3. e) Vuelva a ejecutar la consulta del punto 3(a) y revise el plan de ejecución. ¿Por qué cree que SQL Server no está usando el índice que creó en el punto 3(d)?

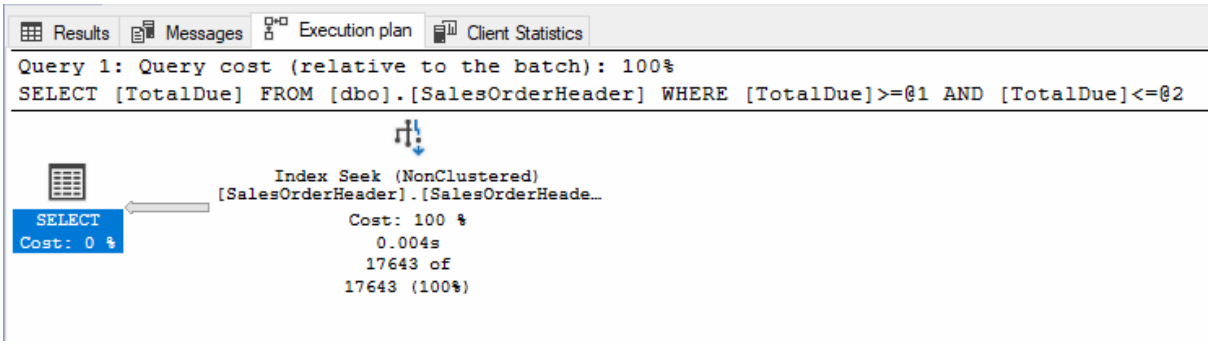


SQL Server no utiliza el índice porque el índice no abarca a todos los atributos del select entonces no se sacaría provecho del índice, conjuntamente, el factor de selectividad entra en juego ya que se recupera más del 50% de las filas totales de la tabla.

Ejercicio 3. f) Ahora cambie la consulta 3(a) para que retorne TotalDue en lugar de todos los atributos de la tabla, y ejecútela:

```
SELECT TotalDue FROM dbo.SalesOrderHeader
WHERE TotalDue BETWEEN 500 AND 40000;
```

Ejercicio 3. g) Revise el plan de ejecución para la consulta del punto 3(f). ¿Cuáles operadores físicos se están usando? ¿Se está usando el índice creado en el punto 3(d)?

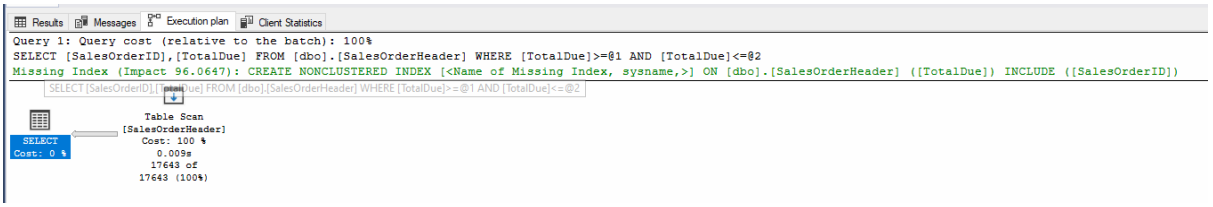


Está utilizando el operador físico Index Seek y sí utiliza el índice creado en el punto 3(d).

Ejercicio 3. h) Cambie la consulta 2(f) para que retorne también SalesOrdeID, y ejecútela:

```
SELECT SalesOrderID, TotalDue
FROM dbo.SalesOrderHeader
WHERE TotalDue BETWEEN 500 AND 40000
```

Ejercicio 3. i) Revise el plan de ejecución para la consulta del punto 3(h). ¿Por qué cree que SQL Server no está usando el índice en este caso? ¿Le da SQL Server alguna sugerencia de índice para optimizar la consulta (missing index)?



SQL Server no utiliza el índice por la misma causa que la del punto 3(e).

Sugiere utilizar un índice para el atributo TotalDue y SalesOrderID.

```
create index SalesOrderHeader_TotalDue_SalesOrderID on
dbo.SalesOrderHeader(TotalDue) include (SalesOrderID)
```

Ejercicio 3. j) Elimine el índice creado en el punto 3(d) y en cree otro nuevo según lo sugerido por SQL Server. ¿Cuál es la diferencia entre el índice original y el sugerido por SQL Server?

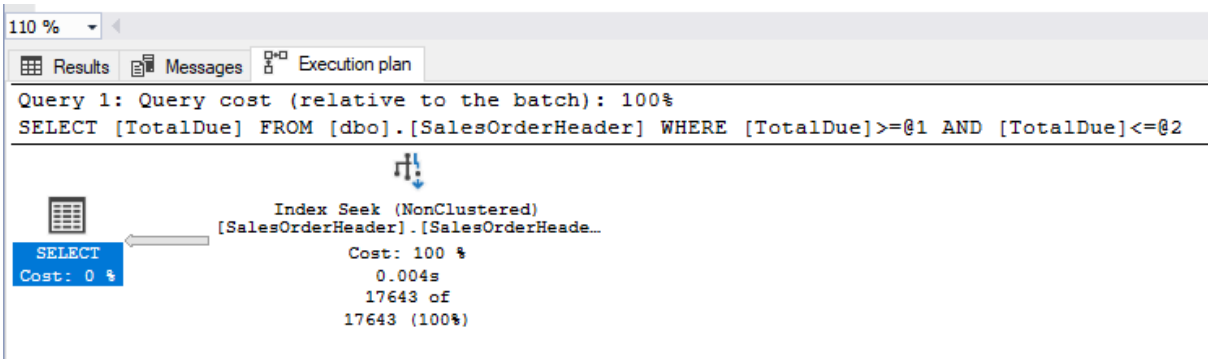
Comandos ejecutados:

```
drop index SalesOrderHeader_OrderDate on
dbo.SalesOrderHeader
```

```
create index SalesOrderHeader_TotalDue_SalesOrderID on
dbo.SalesOrderHeader(TotalDue) include (SalesOrderID)
```

El índice original solo tiene la columna de TotalDue mientras que el índice sugerido contiene las columnas de TotalDue (ordenado según este) y SalesOrderID. Esto permite mejorar el acceso a los datos de estas dos columnas en algunas situaciones.

Ejercicio 3. k) Vuelva a ejecutar la consulta 3(f) y revise el plan de ejecución. ¿Cuáles operadores físicos se están usando? ¿Se está usando el nuevo índice del punto 3(j)? Explique.

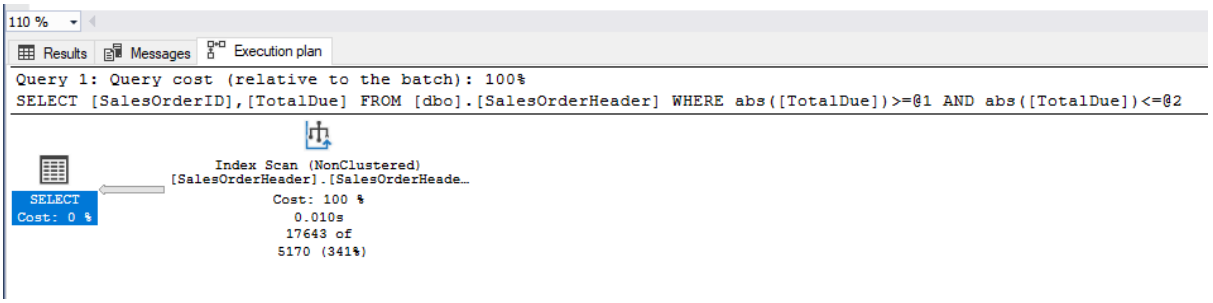


Se está utilizando el operador físico index seek por lo que si se usa el índice del punto 3(j).

Ejercicio 3. l) Cambie la consulta 3(h) para que use la función valor absoluto, y ejecútela.

```
SELECT SalesOrderID, TotalDue
FROM dbo.SalesOrderHeader
WHERE ABS(TotalDue) BETWEEN 500 AND 40000
```

Ejercicio 3. m) Revise el plan de ejecución para la consulta del punto 3(l). ¿Está SQL Server usando el operador Index Scan o el Index Seek sobre el índice? Compare este operador contra el operador que mostraba el plan de ejecución del punto 3(k).



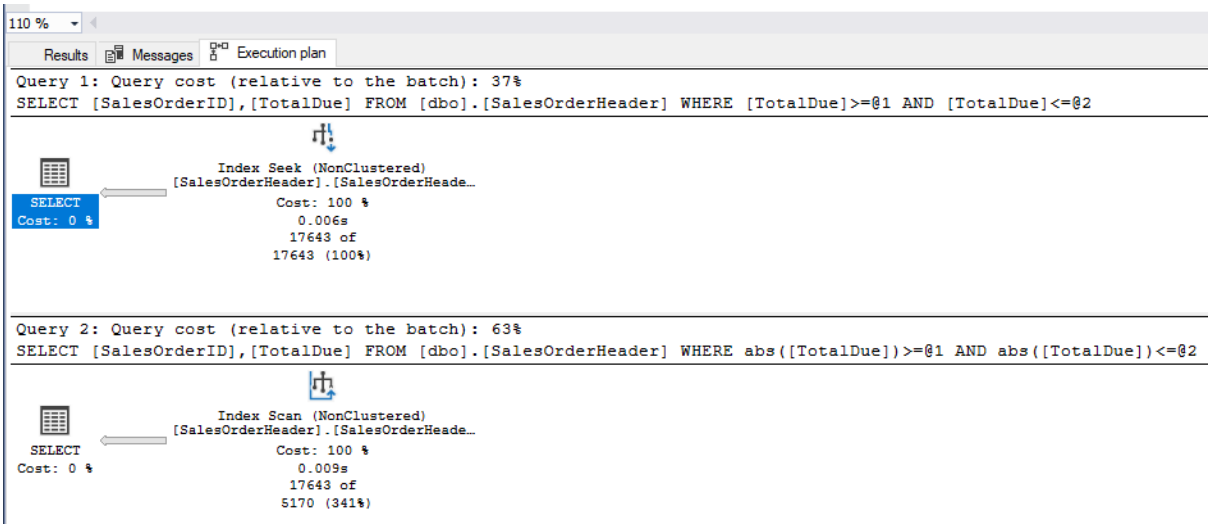
Se utilizó el operador físico index scan a diferencia del punto 3(k) donde se utilizó el index seek.

Ejercicio 3. n) Ejecute los consultas 3(l) y 3(h) juntas, así:

```
SELECT SalesOrderID, TotalDue
FROM dbo.SalesOrderHeader
WHERE TotalDue BETWEEN 500 AND 40000;

SELECT SalesOrderID, TotalDue
FROM dbo.SalesOrderHeader
WHERE ABS(TotalDue) BETWEEN 500 AND 40000;
```

Ejercicio 3. o) Muestre el plan de ejecución real para las consultas del punto 3(n). ¿Cuál es el costo porcentual (relativo al lote o batch) de cada una de las consultas? Reflexione sobre el efecto que tiene el uso de funciones en la optimización de consultas.



El costo porcentual de la primera consulta es del 37% y el del segundo es del 63%. Esto muestra que el uso de funciones aumenta el costo de la consulta.

Ejercicio 3. p) Elimine el índice creado en el punto 3(j).

```
drop index SalesOrderHeader_TotalDue_SalesOrderID on dbo.SalesOrderHeader
```

Ejercicio 4. a) Ejecute las siguientes consultas (marque ambas y ejecútelas juntas):

```
SELECT h.SalesOrderID, d.SalesOrderDetailID, h.SalesPersonID
FROM Sales.SalesOrderHeader h JOIN Sales.SalesOrderDetail d ON d.SalesOrderID = h.SalesOrderID JOIN Sales.SalesPerson p ON p.BusinessEntityID = h.SalesPersonID;
```

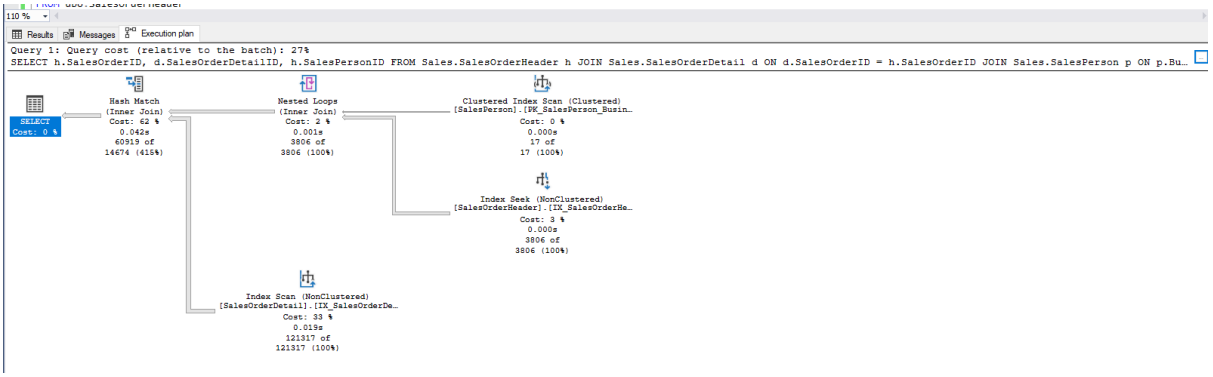
```
SELECT h.SalesOrderID, d.SalesOrderDetailID, h.SalesPersonID
FROM dbo.SalesOrderHeader h JOIN dbo.SalesOrderDetail d ON d.SalesOrderID = h.SalesOrderID JOIN dbo.SalesPerson p ON p.BusinessEntityID = h.SalesPersonID;
```

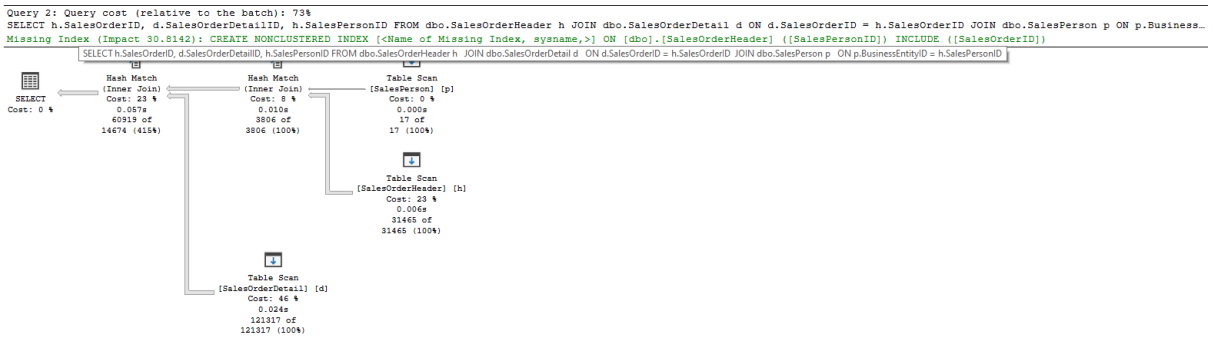
Results | Messages | Execution plan

	SalesOrderID	SalesOrderDetailID	SalesPersonID
60908	71855	112330	290
60909	71857	112353	281
60910	71863	112396	276
60911	71864	112400	275
60912	71889	112723	275
60913	71890	112738	281
60914	71901	112946	275
60915	71914	113083	288
60916	71925	113157	284
60917	71938	113291	282
60918	71941	113327	284
60919	71948	113451	279

	SalesOrderID	SalesOrderDetailID	SalesPersonID
60909	44296	2376	277
60910	44296	2377	277
60911	44296	2378	277
60912	44296	2379	277
60913	44296	2380	277
60914	44297	2381	275
60915	44297	2382	275
60916	44297	2383	275
60917	44297	2384	275
60918	44297	2385	275
60919	44297	2386	275

Ejercicio 4. b) Muestre el plan de ejecución real para la consulta del punto 4(a).





Ejercicio 4. c) ¿Cuál es el costo porcentual (relativo al lote o batch) de cada una de las consultas?

Para la primera consulta el costo porcentual fue de 27% y el de la segunda consulta fue de 73%.

Ejercicio 4. d) ¿A qué atribuye la diferencia entre el costo de estas consultas? Para contestar esta pregunta, analice el plan de ejecución de cada consulta y revise si se usa algún índice y qué tipo de operadores físicos está usando SQL Server en cada caso.

La diferencia entre el costo de las consultas se debe a que la segunda consulta (Utiliza los operadores físicos: hash match y table scan) la hace sobre las tablas del esquema dbo, que no cuenta con índices para optimizar las consultas, mientras que la primera consulta se hace utilizando los índices del esquema sales (Utiliza los operadores físicos: hash match, nested loops, index scan, clustered index scan, index seek).

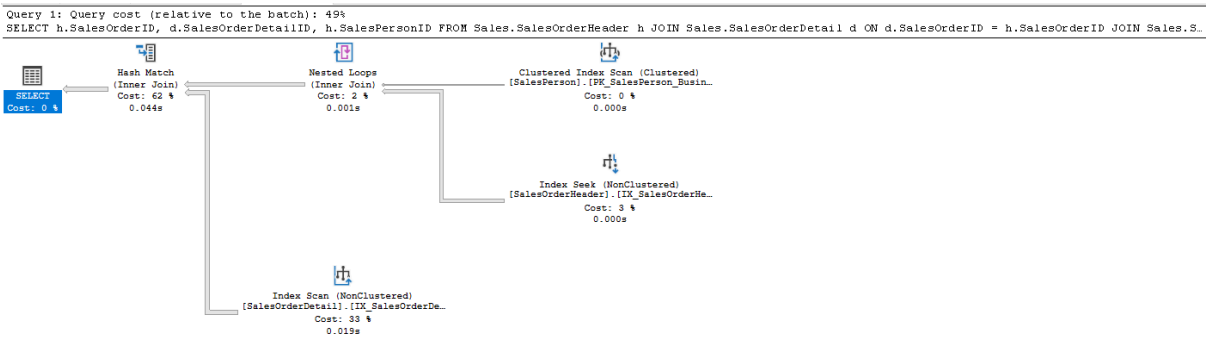
Ejercicio 4. e) Intente mejorar el rendimiento de la segunda consulta del punto 4(a) -la que se hace sobre el esquema dbo- mediante la creación de índices. Se le sugiere revisar la estrategia recomendada en
https://www.toptal.com/sql-server/sql-databasetuning-for-developers
para crear índices cuando hay joins.

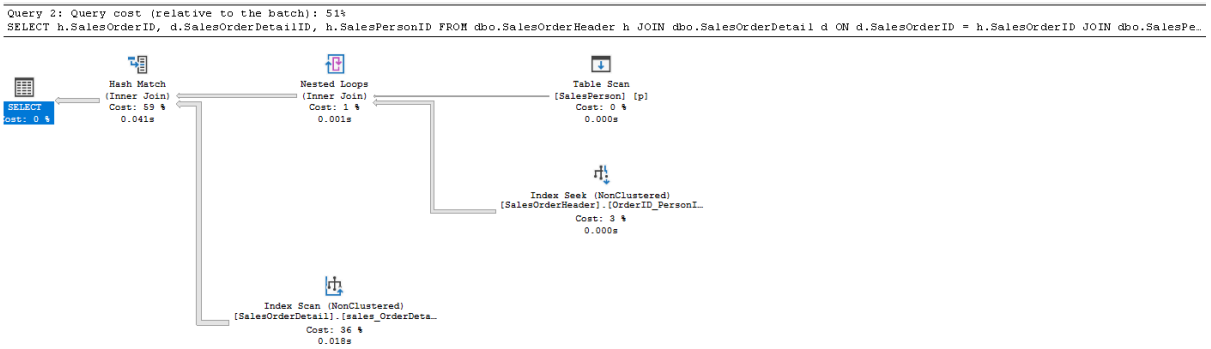
Comandos de creación:

```
create nonclustered index OrderID_PersonID on dbo.SalesOrderHeader (SalesPersonID) include (SalesOrderID)
```

```
create nonclustered index sales_OrderDetailID on [dbo].[SalesOrderDetail] ([SalesOrderID]) include ([SalesOrderDetailID])
```

Ejercicio 4. f) Vuelva a ejecutar las consultas del punto 4(a) y revise el plan de ejecución. ¿Cuál es el costo porcentual (relativo al lote o batch) de cada consulta ahora? ¿Hubo una mejora en el tiempo de ejecución de la segunda consulta? ¿Está SQL Server usando los índices que creó en el punto 4(e)? Explique.





Se muestra una mejora de rendimiento ya que el costo porcentual es de 49% y 51% respectivamente; esto debido a que SQL Server utiliza los índices creados en el punto 4(e), utilizando dos nuevos operadores físicos: nested loops e index seek y deja de utilizar table scan.

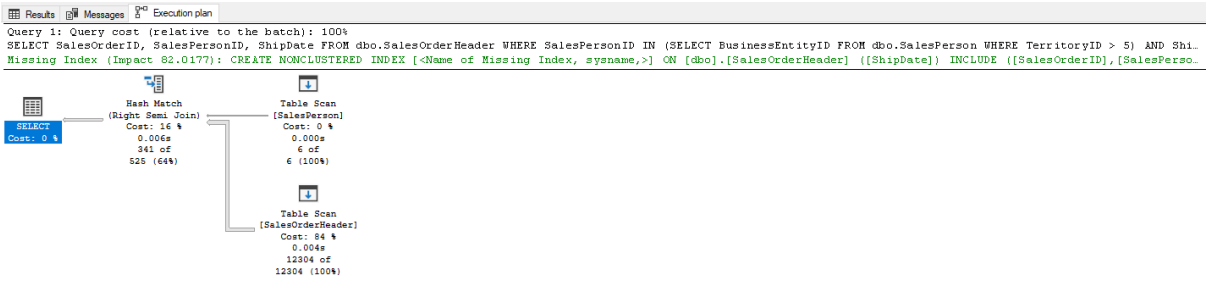
g. Elimine los índices creados en el punto 4(e).

```
drop index OrderID_PersonID on dbo.SalesOrderHeader
drop index sales_OrderDetailID on dbo.SalesOrderDetail
```

Ejercicio 5. a) Ejecute la siguiente consulta:

```
SELECT SalesOrderID, SalesPersonID, ShipDate FROM
dbo.SalesOrderHeader WHERE SalesPersonID IN (SELECT
BusinessEntityID FROM dbo.SalesPerson WHERE TerritoryID > 5)
AND ShipDate > '2014-01-01'
```

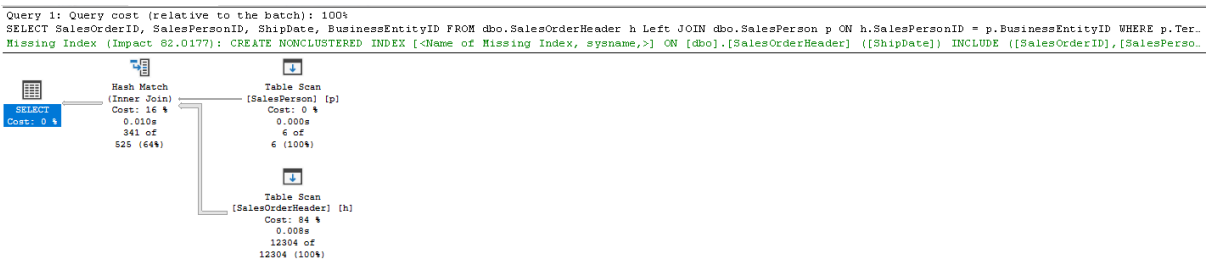
Ejercicio 5. b) Muestre el plan de ejecución real para la consulta del punto 5(a).



Ejercicio 5. c) Re-escriba la consulta 5(a) en otra equivalente que no tenga una consulta anidada, y ejecútela.

```
SELECT SalesOrderID, SalesPersonID, ShipDate, BusinessEntityID
FROM dbo.SalesOrderHeader h JOIN dbo.SalesPerson p ON
h.SalesPersonID = p.BusinessEntityID
WHERE p.TerritoryID > 5 AND ShipDate > '2014-01-01'
```

Ejercicio 5. d) Muestre el plan de ejecución real para la consulta del punto 5(c).



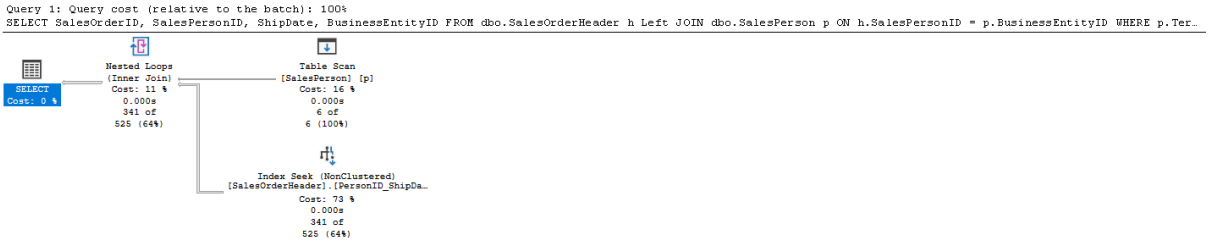
Ejercicio 5. e) Cree uno o varios índices que mejoren el rendimiento de la consulta 5(c).

Comandos de creación:

```
create nonclustered index ShipDate
on dbo.SalesOrderHeader (ShipDate)
include (SalesOrderID,SalesPersonID)
```

```
create nonclustered index PersonID_ShipDate
on dbo.SalesOrderHeader (SalesPersonID,ShipDate)
include (SalesOrderID)
```

Ejercicio 5. f) Vuelva a ejecutar la consulta del punto 5(c) y analice su plan de ejecución. ¿Está SQL Server usando los índices que creó en el punto 5(e)?



SQL Server ahora utiliza los índices creados en el punto 5(e) ya que ahora realiza un index seek y un nested loops en lugar de un table scan.