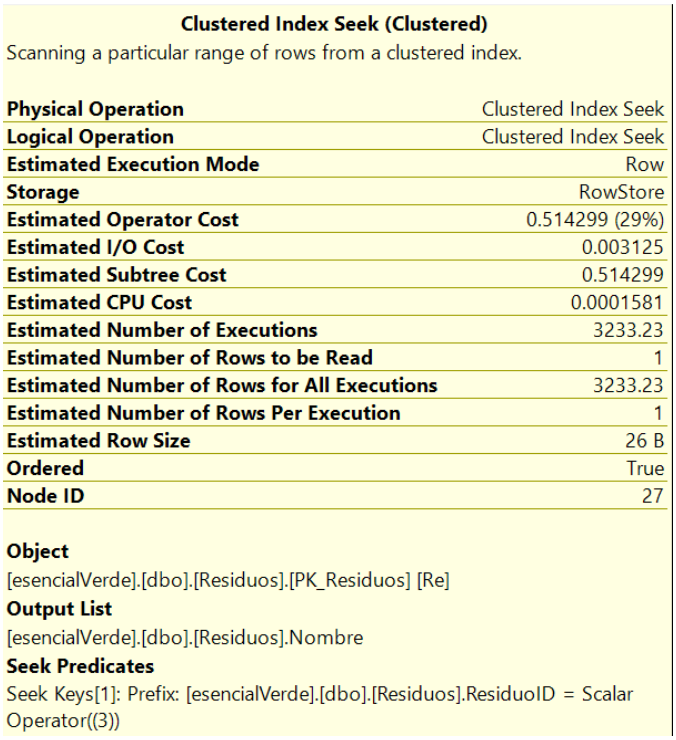
Luis Carlos Navarro Todd c.2022212158
Victoria Sandi Barrantes c.2022146536

```
CREATE VIEW USO_DE_BRANDS AS
SELECT
Res.Nombre AS Residuo,
Brand.Nombre AS Brand_Name,
COUNT(Rlog.AccionRecipienteID) AS Veces_Recogidos,
Trec.Capacidad * SUM(Rlog.CantidadRecipientes) AS  Capacidad_Utilizada,
Unidad.Unidad AS Unidad
FROM RecipientesLogs Rlog
LEFT JOIN TiposRecipientes Trec on Rlog.TipoRecipienteID = Trec.TipoRecipienteID
LEFT JOIN BrandsRecipientes Brand on Brand.BrandRecipienteID =
Trec.BrandRecipienteID
INNER JOIN Residuos Res on Res.ResiduoID = Rlog.ResiduoID
LEFT JOIN UnidadesDeMedidas Unidad on Unidad.UnidadDeMedidaID =
Res.UnidadDeMedidaID
WHERE Rlog.AccionRecipienteID = 3
GROUP BY Res.Nombre, Brand.Nombre, Trec.Capacidad, Unidad.Unidad;
```

Se utiliza una vista dinámica y no en una indexada debido a que en el query se utiliza la tabla RecipientesLogs. Esta tabla lleva registros de los recipientes utilizados por cada local por lo que se ocupan hacer una gran cantidad de inserts en esta tabla. Al utilizar una vista indexada se sacrifica la velocidad de inserción, ya que se tienen que insertar los datos de acuerdo al orden del index de la vista. En este caso, esto empeoraría la calidad de la aplicación. Por esto, se utiliza una vista dinámica. Ya que, a pesar de que la vista va a tardar un poco más, no se sacrifica la eficiencia de inserciones importantes.

```
SELECT Loc.LocalID, Re.Nombre AS Residuo, SUM(CantidadRecipientes) *
TiRec.Capacidad  as Cantidad, AccRec.Descripcion, Brand.Nombre
FROM Locales Loc
RIGHT JOIN RecipientesLogs RLog on Loc.LocalID = RLog.LocalID
LEFT JOIN Residuos Re on Re.ResiduoID = RLog.ResiduoID
LEFT JOIN TiposRecipientes TiRec on TiRec.TipoRecipienteID = RLog.TipoRecipienteID
INNER JOIN AccionesRecipientes AccRec on AccRec.AccionRecipienteID =
RLog.AccionRecipienteID
INNER JOIN BrandsRecipientes Brand on Brand.BrandRecipienteID =
TiRec.BrandRecipienteID
WHERE AccRec.AccionRecipienteID = 3 AND RLog.Hora > CONVERT(datetime2(7),
'2023-06-15 00:00:01')
GROUP BY Loc.LocalID, Re.Nombre, TiRec.Capacidad, AccRec.Descripcion,
Brand.Nombre
EXCEPT
SELECT Loc.LocalID, Re.Nombre AS Residuo, SUM(CantidadRecipientes) *
TiRec.Capacidad  as Cantidad, AccRec.Descripcion, Brand.Nombre
FROM Locales Loc
RIGHT JOIN RecipientesLogs RLog on Loc.LocalID = RLog.LocalID
LEFT JOIN Residuos Re on Re.ResiduoID = RLog.ResiduoID
LEFT JOIN TiposRecipientes TiRec on TiRec.TipoRecipienteID = RLog.TipoRecipienteID
INNER JOIN AccionesRecipientes AccRec on AccRec.AccionRecipienteID =
RLog.AccionRecipienteID
INNER JOIN BrandsRecipientes Brand on Brand.BrandRecipienteID =
TiRec.BrandRecipienteID
WHERE AccRec.AccionRecipienteID = 3 AND RLog.Hora > CONVERT(datetime2(7),
'2023-06-15 00:00:01') AND RLog.ResiduoID = 3
GROUP BY Loc.LocalID, Re.Nombre, TiRec.Capacidad, AccRec.Descripcion,
Brand.Nombre
ORDER BY Loc.LocalID, Cantidad DESC
FOR JSON PATH, ROOT('Base');
```

Duró: 11s



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	0.514299 (29%)
Estimated I/O Cost	0.003125
Estimated Subtree Cost	0.514299
Estimated CPU Cost	0.0001581
Estimated Number of Executions	3233.23
Estimated Number of Rows to be Read	1
Estimated Number of Rows for All Executions	3233.23
Estimated Number of Rows Per Execution	1
Estimated Row Size	26 B
Ordered	True
Node ID	27

Object
[esencialVerde].[dbo].[Residuos].[PK_Residuos] [Re]
Output List
[esencialVerde].[dbo].[Residuos].Nombre
Seek Predicates
Seek Keys[1]: Prefix: [esencialVerde].[dbo].[Residuos].ResiduoID = Scalar Operator((3))

Explicación: Se está utilizando una igualdad en un foreign key en vez de utilizarla en el primary key. *(2do select: WHERE … RLog.ResiduoID = 3)*
Norma: Se deben hacer igualdades en el primary key siempre que sea posible, ya que estos keys están ordenados físicamente

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	0.282635 (23%)
Estimated I/O Cost	0.263866
Estimated Subtree Cost	0.282635
Estimated CPU Cost	0.018769
Estimated Number of Executions	1
Estimated Number of Rows to be Read	16920
Estimated Number of Rows for All Executions	3861.48
Estimated Number of Rows Per Execution	3861.48
Estimated Row Size	33 B
Ordered	False
Node ID	14

Predicate

[esencialVerde].[dbo].[RecipientesLogs].[AccionRecipienteID] as [RLog].[AccionRecipienteID]=(3) AND [esencialVerde].[dbo].[RecipientesLogs].[Hora] as [RLog].[Hora]>'2023-06-15 00:00:01.0000000'

Object

[esencialVerde].[dbo].[RecipientesLogs].[PK_EstadosRecipientesLogs] [RLog]

Output List

[esencialVerde].[dbo].[RecipientesLogs].TipoRecipienteID, [esencialVerde].[dbo].[RecipientesLogs].LocalID, [esencialVerde].[dbo].[RecipientesLogs].CantidadRecipientes, [esencialVerde].[dbo].[RecipientesLogs].ResiduoID

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	0.282635 (23%)
Estimated I/O Cost	0.263866
Estimated Subtree Cost	0.282635
Estimated CPU Cost	0.018769
Estimated Number of Executions	1
Estimated Number of Rows to be Read	16920
Estimated Number of Rows for All Executions	1077.74
Estimated Number of Rows Per Execution	1077.74
Estimated Row Size	33 B
Ordered	False
Node ID	27

Predicate

[esencialVerde].[dbo].[RecipientesLogs].[ResiduoID] as [RLog].[ResiduoID]=(3) AND [esencialVerde].[dbo].[RecipientesLogs].[AccionRecipienteID] as [RLog].[AccionRecipienteID]=(3) AND [esencialVerde].[dbo].[RecipientesLogs].[Hora] as [RLog].[Hora]>'2023-06-15 00:00:01.0000000'

Object

[esencialVerde].[dbo].[RecipientesLogs].[PK_EstadosRecipientesLogs] [RLog]

Output List

[esencialVerde].[dbo].[RecipientesLogs].TipoRecipienteID, [esencialVerde].[dbo].[RecipientesLogs].LocalID, [esencialVerde].[dbo].[RecipientesLogs].CantidadRecipientes

Explicación: El DBMS sugirió que se podía hacer un non-clustered index en la tabla de recipientes logs para optimizar el querie. El index se hace con AccionRecipienteID, ResiduoID y la hora e incluye las columnas de TipoRecipienteID, LocalID y CantidadRecipientes que son tres datos que son utilizados concurrentemente en queries.
Norma: Se pueden utilizar nonclustered indexes en tablas con muchos registros para hacer sus queries de manera más óptima.

```
SELECT Loc.LocalID, Re.Nombre AS Residuo, SUM(CantidadRecipientes) *
TiRec.Capacidad  as Cantidad, AccRec.Descripcion, Brand.Nombre
FROM Locales Loc
RIGHT JOIN RecipientesLogs RLog on Loc.LocalID = RLog.LocalID
LEFT JOIN Residuos Re on Re.ResiduoID = RLog.ResiduoID
LEFT JOIN TiposRecipientes TiRec on TiRec.TipoRecipienteID = RLog.TipoRecipienteID
INNER JOIN AccionesRecipientes AccRec on AccRec.AccionRecipienteID =
RLog.AccionRecipienteID
INNER JOIN BrandsRecipientes Brand on Brand.BrandRecipienteID =
TiRec.BrandRecipienteID
WHERE AccRec.AccionRecipienteID = 3 AND RLog.Hora > CONVERT(datetime2(7),
'2023-06-15 00:00:01')
GROUP BY Loc.LocalID, Re.Nombre, TiRec.Capacidad, AccRec.Descripcion,
Brand.Nombre
EXCEPT
SELECT Loc.LocalID, Re.Nombre AS Residuo, SUM(CantidadRecipientes) *
TiRec.Capacidad  as Cantidad, AccRec.Descripcion, Brand.Nombre
FROM Locales Loc
RIGHT JOIN RecipientesLogs RLog on Loc.LocalID = RLog.LocalID
LEFT JOIN Residuos Re on Re.ResiduoID = RLog.ResiduoID
LEFT JOIN TiposRecipientes TiRec on TiRec.TipoRecipienteID = RLog.TipoRecipienteID
INNER JOIN AccionesRecipientes AccRec on AccRec.AccionRecipienteID =
RLog.AccionRecipienteID
INNER JOIN BrandsRecipientes Brand on Brand.BrandRecipienteID =
TiRec.BrandRecipienteID
WHERE AccRec.AccionRecipienteID = 3 AND RLog.Hora > CONVERT(datetime2(7),
'2023-06-15 00:00:01') AND Re.ResiduoID = 3
GROUP BY Loc.LocalID, Re.Nombre, TiRec.Capacidad, AccRec.Descripcion,
Brand.Nombre
ORDER BY Loc.LocalID, Cantidad DESC
FOR JSON PATH, ROOT('Base');
```

Duró: 1s

CTE’s

Otra herramienta útil a la hora de optimizar los queries es el uso de CTE’s. En este caso, al encapsular el query después del except en un CTE **el tiempo de ejecución bajo a 126 ms.** Por lo que si se tienen selects con un select interno, lo mejor es usar CTE’s.

```
WITH CARGA_LOCAL_POR_BRAND_SOLIDOS AS
(
	SELECT Loc.LocalID, Re.Nombre AS Residuo, SUM(CantidadRecipientes) *
TiRec.Capacidad  as Cantidad, AccRec.Descripcion, Brand.Nombre
	FROM Locales Loc
	RIGHT JOIN RecipientesLogs RLog on Loc.LocalID = RLog.LocalID
	LEFT JOIN Residuos Re on Re.ResiduoID = RLog.ResiduoID
	LEFT JOIN TiposRecipientes TiRec on TiRec.TipoRecipienteID =
RLog.TipoRecipienteID
	INNER JOIN AccionesRecipientes AccRec on AccRec.AccionRecipienteID =
RLog.AccionRecipienteID
	INNER JOIN BrandsRecipientes Brand on Brand.BrandRecipienteID =
TiRec.BrandRecipienteID
	WHERE AccRec.AccionRecipienteID = 3 AND RLog.Hora >
CONVERT(datetime2(7), '2023-06-15 00:00:01') AND Re.ResiduoID = 3
	GROUP BY Loc.LocalID, Re.Nombre, TiRec.Capacidad, AccRec.Descripcion,
Brand.Nombre
)
SELECT Loc.LocalID, Re.Nombre AS Residuo, SUM(CantidadRecipientes) *
TiRec.Capacidad  as Cantidad, AccRec.Descripcion, Brand.Nombre
FROM Locales Loc
RIGHT JOIN RecipientesLogs RLog on Loc.LocalID = RLog.LocalID
LEFT JOIN Residuos Re on Re.ResiduoID = RLog.ResiduoID
LEFT JOIN TiposRecipientes TiRec on TiRec.TipoRecipienteID = RLog.TipoRecipienteID
INNER JOIN AccionesRecipientes AccRec on AccRec.AccionRecipienteID =
RLog.AccionRecipienteID
INNER JOIN BrandsRecipientes Brand on Brand.BrandRecipienteID =
TiRec.BrandRecipienteID
WHERE AccRec.AccionRecipienteID = 3 AND RLog.Hora > CONVERT(datetime2(7),
'2023-06-15 00:00:01')
GROUP BY Loc.LocalID, Re.Nombre, TiRec.Capacidad, AccRec.Descripcion,
Brand.Nombre
EXCEPT
SELECT * FROM CARGA_LOCAL_POR_BRAND_SOLIDOS
ORDER BY Loc.LocalID, Cantidad DESC
FOR JSON PATH, ROOT('Base');
```