Instituto Tecnológico de Costa Rica
Campus Tecnológico Central Cartago
Escuela De Ingeniería En Computación

Bases de datos I

Caso#3 - Preliminar#3

I-Semestre 2023

Jueves 4 mayo

Integrantes:
Kevin Chang - 2022039050
Erika Cerdas - 2022138199

1. Debe ser posible hacer migrations de versiones de bases de datos para controlar lo que se encuentra en desarrollo, qa, stagging y production. se ha dado la directiva de que la creación inicial de la base de datos, todo lo que sea relacionado a creación de tablas irá en un script de ignición o cero, y que posteriormente cada item ya sea función, stored procedure, trigger, views, alters y similares, deberán llevar control de versiones un item por archivo. para simplificar la actualización y controlar las migraciones se va a utilizar [flyway](https://flywaydb.org/).

Flyway Versiones:
al utilizar 'select * from flyway_schema_history' se muestran las diferentes versiones de la base de datos añadida por flyway.

	installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1	1	Creacion0	SQL	V1__Creacion0.sql	1372984850	KEVIN-CC\KEVIN CHANG	2023-05-04 18:18:00.637	339	1
2	2	2	Creacion1	SQL	V2__Creacion1.sql	1440829946	KEVIN-CC\KEVIN CHANG	2023-05-04 18:18:01.253	527	1
3	3	3	InsertData	SQL	V3__InsertData.sql	-1062548611	KEVIN-CC\KEVIN CHANG	2023-05-04 18:18:01.730	437	1
4	4	4	StoredProcedure	SQL	V4__StoredProcedure.sql	-389492334	KEVIN-CC\KEVIN CHANG	2023-05-04 18:18:01.797	36	1
5	5	5	Views	SQL	V5__Views.sql	1211277699	KEVIN-CC\KEVIN CHANG	2023-05-04 18:18:02.123	300	1
6	6	6	InsertData	SQL	V6__InsertData.sql	1085343290	KEVIN-CC\KEVIN CHANG	2023-05-04 18:18:02.207	63	1
7	7	7	ProcedureTVP	SQL	V7__ProcedureTVP.sql	828001412	KEVIN-CC\KEVIN CHANG	2023-05-04 18:18:02.240	18	1

en el report.html también aparecen las versiones disponibles migradas.

Version	Description	Category	Type	Filepath	ExecutionTime
1	Creacion0	Versioned	SQL	V1__Creacion0.sql	00:00.339s
2	Creacion1	Versioned	SQL	V2__Creacion1.sql	00:00.527s
3	InsertData	Versioned	SQL	V3__InsertData.sql	00:00.437s
4	StoredProcedure	Versioned	SQL	V4__StoredProcedure.sql	00:00.036s
5	Views	Versioned	SQL	V5__Views.sql	00:00.300s
6	InsertData	Versioned	SQL	V6__InsertData.sql	00:00.063s
7	ProcedureTVP	Versioned	SQL	V7__ProcedureTVP.sql	00:00.018s

Se pueden ver las diferentes versiones de la base de datos, donde la primera versión es el script de creación y las siguientes versiones son inserts, views y stored procedures entre otros.

2. basado en su diseño, si existe una consulta que requiera al menos 4 joins, cuál opción sería más eficiente: encapsular el query en una vista dinámica o en una vista indexada. Si hay diferencia encontrar una justificación teórica que justifique el hallazgo. la cantidad de datos deben ser lo suficiente para encontrar diferencias.

View con joins original:

```
CREATE VIEW Vista AS
SELECT dbo.contrato.descripcion, dbo.proceso.proceso_id, dbo.proceso.clasificacion,
dbo.desecho_movimientos.posttime, dbo.desecho_movimientos.responsible_name,
dbo.desecho_movimientos.reci_desecho_cantidad,
            dbo.ubicaciones.descripcion AS Expr1, dbo.paises.nombre,
dbo.productores_residuos.nombre AS Expr2, dbo.productores_residuos.porcentaje_carbon,
dbo.productores_residuos.balance
FROM     dbo.contrato INNER JOIN
            dbo.proceso ON dbo.contrato.contrato_id = dbo.proceso.contrato_id INNER
JOIN
            dbo.desecho_movimientos ON dbo.proceso.proceso_id =
dbo.desecho_movimientos.proceso_id INNER JOIN
            dbo.ubicaciones ON dbo.contrato.ubicacion_id = dbo.ubicaciones.ubicacion_id
AND dbo.desecho_movimientos.ubicacion_id = dbo.ubicaciones.ubicacion_id INNER
JOIN
            dbo.paises ON dbo.ubicaciones.pais_id = dbo.paises.pais_id INNER JOIN
            dbo.productores_residuos ON dbo.desecho_movimientos.productor_id =
dbo.productores_residuos.productor_id AND dbo.ubicaciones.ubicacion_id =
dbo.productores_residuos.ubicaicon_id
                                    Where dbo.contrato.contrato_id < 12
```

Indexed View:

```
CREATE VIEW dbo.indexado
WITH SCHEMABINDING
AS
   SELECT dbo.contrato.descripcion, dbo.proceso.proceso_id, dbo.proceso.clasificacion,
dbo.desecho_movimientos.posttime, dbo.desecho_movimientos.responsible_name,
dbo.desecho_movimientos.reci_desecho_cantidad,
            dbo.ubicaciones.descripcion AS Expr1, dbo.paises.nombre,
dbo.productores_residuos.nombre AS Expr2, dbo.productores_residuos.porcentaje_carbon,
dbo.productores_residuos.balance
   FROM     dbo.contrato INNER JOIN
            dbo.proceso ON dbo.contrato.contrato_id = dbo.proceso.contrato_id INNER
JOIN
            dbo.desecho_movimientos ON dbo.proceso.proceso_id =
dbo.desecho_movimientos.proceso_id INNER JOIN
            dbo.ubicaciones ON dbo.contrato.ubicacion_id = dbo.ubicaciones.ubicacion_id
AND dbo.desecho_movimientos.ubicacion_id = dbo.ubicaciones.ubicacion_id INNER
```

```
JOIN
            dbo.paises ON dbo.ubicaciones.pais_id = dbo.paises.pais_id INNER JOIN
            dbo.productores_residuos ON dbo.desecho_movimientos.productor_id =
dbo.productores_residuos.productor_id AND dbo.ubicaciones.ubicacion_id =
dbo.productores_residuos.ubicaicon_id
   WHERE dbo.contrato.contrato_id < 12

GO

CREATE UNIQUE CLUSTERED INDEX IX_Vista ON indexado (responsible_name);
```

Dynamic View:

```
DECLARE @sql NVARCHAR(MAX)

SET @sql = 'CREATE VIEW dinamico AS ' + CHAR(13) +
        'SELECT TOP 100 PERCENT dbo.contrato.descripcion, dbo.proceso.proceso_id,
dbo.proceso.clasificacion, dbo.desecho_movimientos.posttime,
dbo.desecho_movimientos.responsible_name, ' + CHAR(13) +
        'dbo.desecho_movimientos.reci_desecho_cantidad, dbo.ubicaciones.descripcion AS
Expr1, dbo.paises.nombre, dbo.productores_residuos.nombre AS Expr2, ' + CHAR(13) +
        'dbo.productores_residuos.porcentaje_carbon, dbo.productores_residuos.balance' +
CHAR(13) +
        'FROM dbo.contrato' + CHAR(13) +
        'INNER JOIN dbo.proceso ON dbo.contrato.contrato_id = dbo.proceso.contrato_id'
+ CHAR(13) +
        'INNER JOIN dbo.desecho_movimientos ON dbo.proceso.proceso_id =
dbo.desecho_movimientos.proceso_id' + CHAR(13) +
        'INNER JOIN dbo.ubicaciones ON dbo.contrato.ubicacion_id =
dbo.ubicaciones.ubicacion_id AND dbo.desecho_movimientos.ubicacion_id =
dbo.ubicaciones.ubicacion_id' + CHAR(13) +
        'INNER JOIN dbo.paises ON dbo.ubicaciones.pais_id = dbo.paises.pais_id' +
CHAR(13) +
        'INNER JOIN dbo.productores_residuos ON
dbo.desecho_movimientos.productor_id = dbo.productores_residuos.productor_id AND
dbo.ubicaciones.ubicacion_id = dbo.productores_residuos.ubicaicon_id' + CHAR(13) +
        'WHERE dbo.contrato.contrato_id < 12' + CHAR(13) +
        'ORDER BY dbo.productores_residuos.balance'

EXEC sp_executesql @sql;
```

Tomando en cuenta el view dinámico y el view indexado se pueden notar leves diferencias. Podemos ver estas diferencias en eficiencia a través de set statistics y el estimated execution plan.

Statistics:

View Dinámico

```
(21300 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'desecho_movimientos'. Scan count 1, logical reads 349, physical reads 1, page server reads 0, read-ahead reads 340, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'paises'. Scan count 0, logical reads 4, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'ubicaciones'. Scan count 0, logical reads 4, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'productores_residuos'. Scan count 1, logical reads 2, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'contrato'. Scan count 0, logical reads 4, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'proceso'. Scan count 1, logical reads 2, physical reads 1, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
```

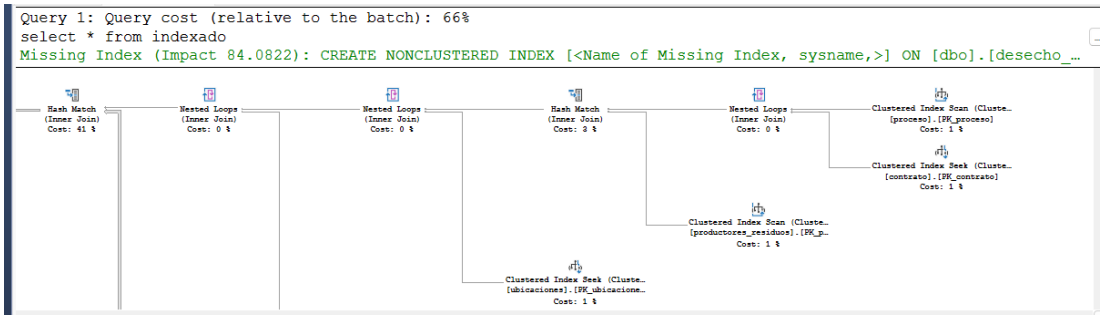
View Indexado:

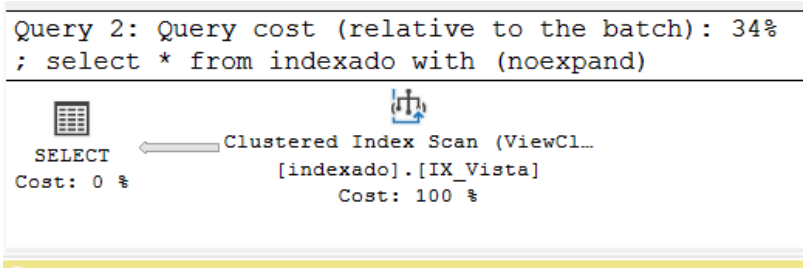
```
(21300 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'desecho_movimientos'. Scan count 1, logical reads 349, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'paises'. Scan count 0, logical reads 4, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'ubicaciones'. Scan count 0, logical reads 4, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
```

```
Table 'productores_residuos'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'contrato'. Scan count 0, logical reads 4, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
Table 'proceso'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead reads 0, page server read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob page server reads 0, lob read-ahead reads 0, lob page server read-ahead reads 0.
```

Se puede ver que los statistics de los views muestran la cantidad de los diferentes tipos de reads para cada view. El indexed posee 368 reads mientras que el dinámico posee 705 reads. Esto muestra que el dinámico posee mayor cantidad de procesos o acciones lo cual lo hace menos eficiente que el indexado. En este caso en específico, la vista indexada es más eficiente. Las vistas dinámicas sirven mejor con vistas de menor tamaño.

Ya que una vista indexada está guardada en un índice en la base de datos y una dinámica solo se corre actualmente cuando se necesita. La vista indexada es más eficiente para grandes cantidades de datos como se tiene en las tablas utilizadas o vistas más complejas como lo es el view con 4 joins que se usa. Esto ocurre ya que el indexed view tiene que leer menos datos en las tablas.





Estas imágenes muestran el estimated execution plan de él view antes y después de indexar. Indexar la vista hace más eficiente la vista.

La cantidad de datos utilizados no mostró cambios relevantes entre el estimated execution plan del indexed y el dynamic view.

3. determinar una norma de estrategia de optimización para su diseño de base de datos, determinar una consulta real del sistema que contenga todos los componentes comunes de un query: fields, joins, left/right join, aggregate functions, except/intersect, group by, sort, for json, wheres sobre campos primary y non primary, igualdades y desigualdades. retornando una cantidad generosa de registros evalúe tiempos de ejecución y plan de ejecución de la consulta, y con ello diseñe un conjunto de pasos o normas, que debe seguir el equipo de desarrollo para garantizar que las consultas complejas se optimicen de una forma estándar y ordenada para la organización. Justifique cada normal con scripts ejemplos para hacer la demostración en tiempo real.

Script Original	Script Optimizado
SELECT dbo.contrato.descripcion, dbo.proceso.proceso_id, dbo.proceso.clasificacion, dbo.desecho_movimientos.posttime, dbo.desecho_movimientos.responsible_name, dbo.desecho_movimientos.reci_desecho_cantidad, dbo.ubicaciones.descripcion AS Expr1, dbo.paises.nombre, dbo.productores_residuos.nombre AS Expr2, dbo.productores_residuos.porcentaje_carbon, dbo.productores_residuos.balance, COUNT(*) AS total, SUM(dbo.desecho_movimientos.reci_desecho_cantidad) AS total_desechos, MIN(dbo.desecho_movimientos.posttime) AS fecha_minima, MAX(dbo.desecho_movimientos.posttime) AS fecha_maxima FROM dbo.contrato INNER JOIN dbo.proceso ON dbo.contrato.contrato_id = dbo.proceso.contrato_id INNER JOIN	CREATE NONCLUSTERED INDEX idx_posttime ON dbo.desecho_movimientos (posttime); CREATE NONCLUSTERED INDEX idx_ubicacion_id ON dbo.contrato (ubicacion_id); SELECT dbo.contrato.descripcion, dbo.proceso.proceso_id, dbo.proceso.clasificacion, dbo.desecho_movimientos.posttime, dbo.desecho_movimientos.responsible_name, dbo.desecho_movimientos.reci_desecho_cantidad, dbo.ubicaciones.descripcion AS Expr1, dbo.paises.nombre, dbo.productores_residuos.nombre AS Expr2, dbo.productores_residuos.porcentaje_carbon, dbo.productores_residuos.balance, COUNT(*) AS total, SUM(dbo.desecho_movimientos.reci_desecho_cantidad) AS total_desechos, MIN(dbo.desecho_movimientos.posttime) AS fecha_minima,

```
dbo.desecho_movimientos
		ON dbo.proceso.proceso_id
= dbo.desecho_movimientos.proceso_id
	INNER JOIN dbo.ubicaciones
		ON
dbo.contrato.ubicacion_id =
dbo.ubicaciones.ubicacion_id
		AND
dbo.desecho_movimientos.ubicacion_id =
dbo.ubicaciones.ubicacion_id
	INNER JOIN dbo.paises
		ON dbo.ubicaciones.pais_id
= dbo.paises.pais_id
	INNER JOIN
dbo.productores_residuos
		ON
dbo.desecho_movimientos.productor_id =
dbo.productores_residuos.productor_id
		AND
dbo.ubicaciones.ubicacion_id =
dbo.productores_residuos.ubicaicon_id
WHERE dbo.contrato.contrato_id < 12
	AND
dbo.desecho_movimientos.reci_desecho_ca
ntidad > 0
	AND
dbo.desecho_movimientos.posttime
BETWEEN '2022-01-01' AND '2022-12-31'
	AND
dbo.productores_residuos.balance < 0
GROUP BY
	dbo.contrato.descripcion,
	dbo.proceso.proceso_id,
	dbo.proceso.clasificacion,
	dbo.desecho_movimientos.posttime,

dbo.desecho_movimientos.responsible_nam
e,

dbo.desecho_movimientos.reci_desecho_ca
ntidad,
	dbo.ubicaciones.descripcion,
	dbo.paises.nombre,
	dbo.productores_residuos.nombre,

dbo.productores_residuos.porcentaje_carbo
n,
	dbo.productores_residuos.balance
HAVING COUNT(*) > 1
```

```
MAX(dbo.desecho_movimientos.posttime)
AS fecha_maxima
FROM dbo.contrato
	INNER JOIN dbo.proceso
		ON dbo.contrato.contrato_id
= dbo.proceso.contrato_id
	INNER JOIN
dbo.desecho_movimientos
		ON dbo.proceso.proceso_id
= dbo.desecho_movimientos.proceso_id
	INNER JOIN dbo.ubicaciones
		ON
dbo.contrato.ubicacion_id =
dbo.ubicaciones.ubicacion_id
		AND
dbo.desecho_movimientos.ubicacion_id =
dbo.ubicaciones.ubicacion_id
	INNER JOIN dbo.paises
		ON dbo.ubicaciones.pais_id
= dbo.paises.pais_id
	INNER JOIN
dbo.productores_residuos
		ON
dbo.desecho_movimientos.productor_id =
dbo.productores_residuos.productor_id
		AND
dbo.ubicaciones.ubicacion_id =
dbo.productores_residuos.ubicaicon_id
WHERE dbo.contrato.contrato_id < 12
	AND
dbo.desecho_movimientos.reci_desecho_ca
ntidad > 0
	AND
dbo.desecho_movimientos.posttime
BETWEEN '2022-01-01' AND '2022-12-31'
	AND
dbo.productores_residuos.balance < 0
GROUP BY
	dbo.contrato.descripcion,
	dbo.proceso.proceso_id,
	dbo.proceso.clasificacion,
	dbo.desecho_movimientos.posttime,

dbo.desecho_movimientos.responsible_nam
e,

dbo.desecho_movimientos.reci_desecho_ca
ntidad,
	dbo.ubicaciones.descripcion,
```

ORDER BY dbo.contrato.descripcion, dbo.desecho_movimientos.posttime DESC FOR JSON AUTO;	dbo.paises.nombre, dbo.productores_residuos.nombre, dbo.productores_residuos.porcentaje_carbon, dbo.productores_residuos.balance HAVING COUNT(*) > 1 ORDER BY dbo.contrato.descripcion, dbo.desecho_movimientos.posttime DESC FOR JSON AUTO;

Unidad de workload	Explicación	Norma	Optimizado
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** 0.294787 (78%) **Estimated I/O Cost** 0.259421 **Estimated Subtree Cost** 0.294787 **Estimated CPU Cost** 0.0353658 **Estimated Number of Executions** 1 **Estimated Number of Rows to be Read** 32008 **Estimated Number of Rows for All Executions** 1 **Estimated Number of Rows Per Execution** 1 **Estimated Row Size** 49 B **Ordered** False **Node ID** 9 **Predicate** [case3].[dbo].[desecho_movimientos].[reci_desecho_cantidad]>(0.00) AND [case3].[dbo].[desecho_movimientos].[posttime]>='2022-01-01 00:00:00.000' AND [case3].[dbo].[desecho_movimientos].[posttime] <='2022-12-31 00:00:00.000' **Object** [case3].[dbo].[desecho_movimientos].[PK_desecho_movimientos] **Output List** [case3].[dbo].[desecho_movimientos].posttime, [case3].[dbo].[desecho_movimientos].responsible_name, [case3].[dbo].[desecho_movimientos].ubicacion_id, [case3].[dbo].[desecho_movimientos].productor_id, [case3].[dbo].[desecho_movimientos].reci_desecho_cantidad, [case3].[dbo].[desecho_movimientos].proceso_id	Acá se está recorriendo la tabla de desecho_movimientos con base en el PK_desecho_movimientos. A partir de esto, se extraen los registros que se encuentren en el rango de fechas dado	-Ordenar la tabla de desecho_movimientos por fechas, esto para facilitar la búsqueda por fechas sin tener que revisar toda la tabla -Indexar el posttime	**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** 0.0032831 (6%) **Estimated I/O Cost** 0.003125 **Estimated Subtree Cost** 0.0032831 **Estimated CPU Cost** 0.0001581 **Estimated Number of Executions** 1 **Estimated Number of Rows to be Read** 1 **Estimated Number of Rows for All Executions** 1 **Estimated Number of Rows Per Execution** 1 **Estimated Row Size** 17 B **Ordered** True **Node ID** 10 **Object** [case3].[dbo].[desecho_movimientos].[idx_posttime] **Output List** [case3].[dbo].[desecho_movimientos].posttime, [case3].[dbo].[desecho_movimientos].w_mov_id **Seek Predicates** Seek Keys[1]: Start: [case3].[dbo].[desecho_movimientos].posttime >= Scalar Operator('2022-01-01 00:00:00.000'), End: [case3].[dbo].[desecho_movimientos].posttime <= Scalar Operator('2022-12-31 00:00:00.000')
Hash Match Use each row from the top input to build a hash table, and each row from the bottom input to probe into the hash table, outputting all matching rows. **Physical Operation** Hash Match **Logical Operation** Inner Join **Estimated Execution Mode** Row **Estimated Operator Cost** 0.0555568 (15%) **Estimated I/O Cost** 0 **Estimated Subtree Cost** 0.353628 **Estimated CPU Cost** 0.0177847 **Estimated Number of Executions** 1 **Estimated Number of Rows Per Execution** 1 **Estimated Number of Rows for All Executions** 1 **Estimated Row Size** 91 B **Node ID** 8 **Output List** [case3].[dbo].[contrato].contrato_id, [case3].[dbo].[contrato].descripcion, [case3].[dbo].[desecho_movimientos].posttime, [case3].[dbo].[desecho_movimientos].responsible_name, [case3].[dbo].[desecho_movimientos].ubicacion_id, [case3].[dbo].[desecho_movimientos].productor_id, [case3].[dbo].[desecho_movimientos].reci_desecho_cantidad, [case3].[dbo].[desecho_movimientos].proceso_id **Hash Keys Probe** [case3].[dbo].[contrato].ubicacion_id	El engine está poniendo en el hash la llave ubicacion_id, en cada celda el desecho_movimientos recorre cada contrato y para cada contrato le hace hash a la ubicacion_id para sacar el número y descripción de contrato	-Indexar la llave foránea de la tabla que usa para hacer el probe	**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** 0.0032831 (9%) **Estimated I/O Cost** 0.003125 **Estimated Subtree Cost** 0.0032831 **Estimated CPU Cost** 0.0001581 **Estimated Number of Executions** 1 **Estimated Number of Rows to be Read** 1 **Estimated Number of Rows for All Executions** 1 **Estimated Number of Rows Per Execution** 1 **Estimated Row Size** 9 B **Ordered** True **Node ID** 14 **Object** [case3].[dbo].[contrato].[idx_ubicacion_id] **Output List** [case3].[dbo].[contrato].contrato_id **Seek Predicates** Seek Keys[1]: Prefix: [case3].[dbo].[contrato].ubicacion_id = Scalar Operator([case3].[dbo].[desecho_movimientos].[ubicacion_id]), End: [case3].[dbo].[contrato].contrato_id < Scalar Operator((12))

4. simplifique por medio de un [CTE] (https://learn.microsoft.com/en-us/sql/t-sql/queries/with-common-table-expression-transact-sql?view=sql-server-2017) la consulta más optimizada del ejercicio de la norma de optimización y averigue si existe diferencia sustancial de rendimiento para que esto también sea agregado a la norma del equipo de desarrollo

CTE del Query de Optimizado:

```
WITH CTE_contracts AS (
    SELECT *
    FROM dbo.contrato
    WHERE contrato_id < 12
), CTE_movements AS (
    SELECT *
    FROM dbo.desecho_movimientos
    WHERE reci_desecho_cantidad > 0
        AND posttime BETWEEN '2022-01-01' AND '2022-12-31'
), CTE_producers AS (
    SELECT *
    FROM dbo.productores_residuos
    WHERE balance < 0
)
SELECT
    CTE_contracts.descripcion,
    dbo.proceso.proceso_id,
    dbo.proceso.clasificacion,
    CTE_movements.posttime,
    CTE_movements.responsible_name,
    CTE_movements.reci_desecho_cantidad,
    dbo.ubicaciones.descripcion AS Expr1,
    dbo.paises.nombre,
    CTE_producers.nombre AS Expr2,
    CTE_producers.porcentaje_carbon,
    CTE_producers.balance,
    COUNT(*) AS total,
    SUM(CTE_movements.reci_desecho_cantidad) AS total_desechos,
    MIN(CTE_movements.posttime) AS fecha_minima,
    MAX(CTE_movements.posttime) AS fecha_maxima
FROM CTE_contracts
        INNER JOIN dbo.proceso
                ON CTE_contracts.contrato_id = dbo.proceso.contrato_id
        INNER JOIN CTE_movements
```

```
            ON dbo.proceso.proceso_id = CTE_movements.proceso_id
        INNER JOIN dbo.ubicaciones
            ON CTE_contracts.ubicacion_id = dbo.ubicaciones.ubicacion_id
            AND CTE_movements.ubicacion_id = dbo.ubicaciones.ubicacion_id
        INNER JOIN dbo.paises
            ON dbo.ubicaciones.pais_id = dbo.paises.pais_id
        INNER JOIN CTE_producers
            ON CTE_movements.productor_id = CTE_producers.productor_id
            AND dbo.ubicaciones.ubicacion_id = CTE_producers.ubicaicon_id
GROUP BY
   CTE_contracts.descripcion,
   dbo.proceso.proceso_id,
   dbo.proceso.clasificacion,
   CTE_movements.posttime,
   CTE_movements.responsible_name,
   CTE_movements.reci_desecho_cantidad,
   dbo.ubicaciones.descripcion,
   dbo.paises.nombre,
   CTE_producers.nombre,
   CTE_producers.porcentaje_carbon,
   CTE_producers.balance
HAVING COUNT(*) > 1
ORDER BY
   CTE_contracts.descripcion,
   CTE_movements.posttime DESC
FOR JSON AUTO;
```

La diferencia entre el query creado al aplicar las normas y su versión utilizando CTE es mínima y no sustancial. Las diferencias de costo de ejecución eran muy similares entre estas dos.