Práctica 3

E01

¿Alguna diferencia entre las dos ejecuciones? ¿Y entre las dos consultas? ¿Por qué?

La diferencia entre las dos ejecuciones es mínima, esto se debe a que hemos limpiado la cache con:
DBCC FREEPROCCACHE WITH NO_INFOMSGS;
DBCC DROPCLEANBUFFERS WITH NO_INFOMSGS;
Si no utilizamos estas dos líneas se vería como la segunda ejecución sería más rápida que la primera.
Se puede observar que la recuperación de datos de la tabla usuario es más lenta frente a la tabla usuarioH debido a que usuario es una tabla más compleja con índices y constraints.
Aunque si lo que solicitamos fuera un select * la tabla usuario sería más rápida debido al uso de índices.

1. Ejecución:

```
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 1 ms.

(1 row affected)

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 78 ms, tiempo transcurrido = 349 ms.

(1 row affected)

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 62 ms, tiempo transcurrido = 376 ms.

Completion time: 2021-10-15T15:21:15.6221163+02:00
```

2. Ejecución:

```
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 1 ms.

(1 row affected)

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 78 ms, tiempo transcurrido = 348 ms.

(1 row affected)

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 94 ms, tiempo transcurrido = 392 ms.

Completion time: 2021-10-15T15:24:10.2575793+02:00
```

E02

¿Alguna diferencia? ¿Por qué?

Como comente en el ejercicio anterior, aquí se puede apreciar una gran diferencia en el tiempo de recuperación de filas de las diferentes tablas.
UsuarioH es más lenta comparada con Usuario debido al uso de índices de la tabla Usuario, que le proporciona mayor rapidez en el acceso y recuperación de filas.

1.Ejecución

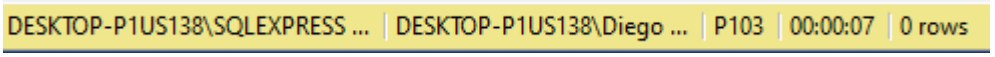
```
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 1 ms.
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

(1 row affected)

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 78 ms, tiempo transcurrido = 351 ms.
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

(1 row affected)

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.
```

E03

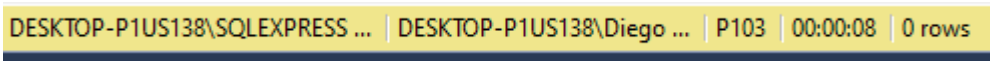
¿Alguna diferencia?¿Por qué?

La inserción de datos es más lenta para la tabla usuario que para usuarioH, esto se debe al número de índices que tiene la tabla Usuario, cuanto mayor sea el número de índices más se verá afectado el rendimiento de la tabla a la hora de hacer inserts.

1. Ejecución



2. Ejecución



E03b

¿Alguna diferencia?¿Por qué?

Ahora la tabla usuario es más rápida que usuarioH. Al contrario que con el insert el comando delete trabaja más rápido cuanto mayor sea el número de índices de la tabla.
Ya que para borrar las filas, internamente trabaja como un Select y después borra los registros identificados.

1. Ejecución

```
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 219 ms, tiempo transcurrido = 1474 ms.
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 750 ms, tiempo transcurrido = 944 ms.

Completion time: 2021-10-15T16:14:52.8343392+02:00
```

E04

USE P103 go select count(*) from comentario where respondea is not null

```
CREATE NONCLUSTERED INDEX comentarioConRespondea
    ON comentario  (respondea)
    WHERE respondea IS NOT NULL ;
GO
```

E04b

Define otro índice filtrado para select usuId, respondea from comentario where respondea is not null

```
CREATE NONCLUSTERED INDEX comentarioConRespondeaUsuId
    ON comentario  (usuId,respondea)
    WHERE respondea IS NOT NULL ;
GO
```

E05

¿Qué va a hacer el servidor, en qué ha cambiado el plan?

Mientras que antes hacía un scan de ambas tablas, ahora hace un scan de comentario y utiliza el índice de usuario, finalmente hace un ‘join’ de ambas tablas esta vez con ‘nested loops’ y muestra los datos.

E06

¿Qué ha ocurrido?

Al crear un índice para cada tabla y después hacer un ‘select’ de ambas tablas, se accede a los dos índices para reducir el tiempo al mostrar el ‘join’ mediante ‘nested loops’.

E07

Compara el plan anterior con el de la siguiente consulta:

En este ejemplo primero se accede al índice de comentario, después se ordena al mismo tiempo que se accede al índice de usuario y finalmente se mezclan las tablas con ‘nested loops’ y se muestran.

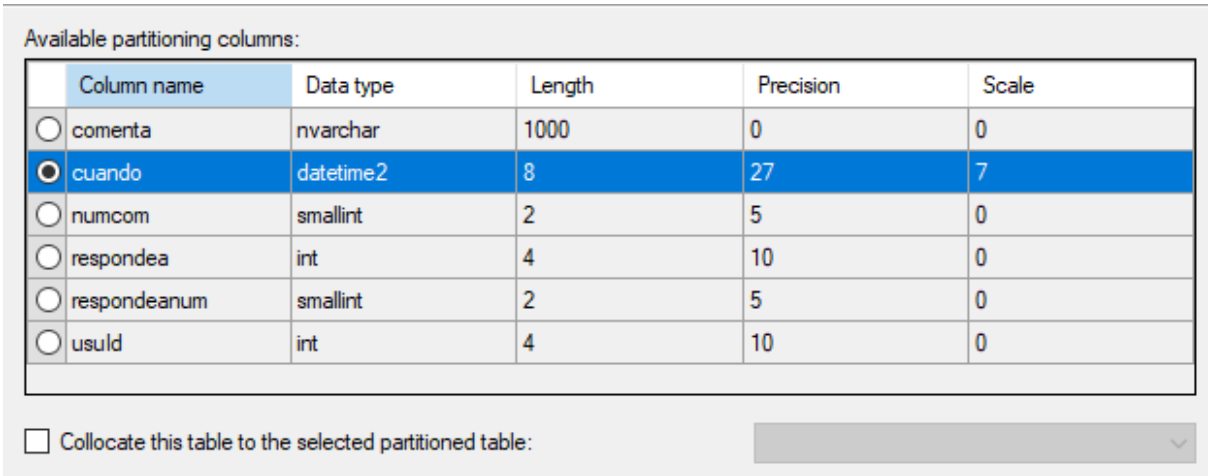
E08

Realiza la partición horizontal de la tabla COMENTARIO en otra base de datos que crees para la ocasión. Compara tiempos de ejecución entre la tabla original y la particionada.

Creamos los filegroups a la nueva base de datos.

```
ALTER DATABASE pruebaE08
ADD FILEGROUP January
GO
ALTER DATABASE pruebaE08
ADD FILEGROUP February
GO
ALTER DATABASE pruebaE08
ADD FILEGROUP March
GO
ALTER DATABASE pruebaE08
ADD FILEGROUP April
GO
ALTER DATABASE pruebaE08
ADD FILEGROUP May
GO
ALTER DATABASE pruebaE08
ADD FILEGROUP June
GO
ALTER DATABASE pruebaE08
ADD FILEGROUP July
GO
ALTER DATABASE pruebaE08
ADD FILEGROUP Avgust
GO
ALTER DATABASE pruebaE08
ADD FILEGROUP September
GO
ALTER DATABASE pruebaE08
```

Creando la partición en una nueva base de datos.



Script generado

```
USE [pruebaE08]
GO
BEGIN TRANSACTION
CREATE PARTITION FUNCTION [comentariosMeses](datetime2(7)) AS RANGE LEFT FOR VALUES (N'2017-01-01T00:00:00', N'2017-02-01T00:00:00',


CREATE PARTITION SCHEME [comentariosMesesScheme] AS PARTITION [comentariosMeses] TO ([January], [February], [March], [April], [May],



CREATE CLUSTERED INDEX [ClusteredIndex_on_comentariosMesesScheme_637699168702979762] ON [dbo].[comentarioE8]
(
    [cuando]
)WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF) ON [comentariosMesesScheme]([cuando])


DROP INDEX [ClusteredIndex_on_comentariosMesesScheme_637699168702979762] ON [dbo].[comentarioE8]




COMMIT TRANSACTION
```

Se ejecutan las siguiente consultas

```
USE pruebaE08
DBCC FREEPROCCACHE WITH NO_INFOMSGS
DBCC DROPCLEANBUFFERS WITH NO_INFOMSGS; -- necesitamos limpiar caché para comparar
SET STATISTICS TIME ON;
select * from comentarioE8 where cuando BETWEEN '1/1/2017' AND '1/2/2017';
SET STATISTICS TIME OFF;

use p103
DBCC FREEPROCCACHE WITH NO_INFOMSGS
DBCC DROPCLEANBUFFERS WITH NO_INFOMSGS; -- necesitamos limpiar caché para comparar
SET STATISTICS TIME ON;
select * from comentario where cuando BETWEEN '1/1/2017' AND '1/2/2017';
SET STATISTICS TIME OFF;
```

Se obtienen los siguientes resultados

Tabla particionada:

```
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 282 ms, tiempo transcurrido = 1225 ms.

Completion time: 2021-10-15T18:03:08.0957852+02:00
```

Tabla no particionada:

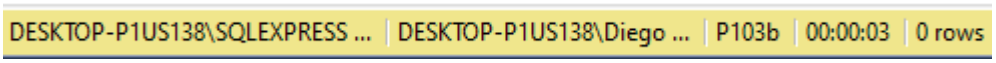
```
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 593 ms, tiempo transcurrido = 2443 ms.

Completion time: 2021-10-15T18:01:28.6050712+02:00
```

Se puede apreciar en los resultados que el acceso a los datos en las tablas particionadas es más rápido que en las no particionadas, tanto es así que el tiempo de ejecución para la tabla no particionada ha sido el doble.

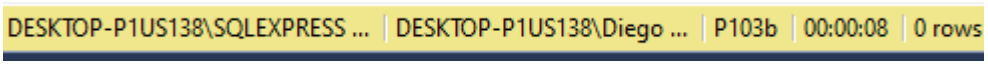
E9

¿Qué ha llevado menos tiempo?

1.Ejecución



2.Ejecución



Se ha comprobado que la tabla que se ha almacenado en memoria RAM va mas rapida que la que no lo esta.

E10

Planteada la consulta siguiente, establece los índices necesarios para acelerar la respuesta:

```
select respondea,respondeanum,cuando
from comentario
where respondea is not null and cuando='2017-12-19'
```

Sin índices:

```
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

(299 rows affected)

(1 row affected)

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 328 ms, tiempo transcurrido = 1939 ms.

Completion time: 2021-10-15T18:44:32.8819925+02:00
```

Con el siguiente índice:

```
CREATE NONCLUSTERED INDEX respondeCuando
ON comentario (respondea,cuando)
WHERE respondea is not null
GO
```

```
Tiempo de análisis y compilación de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

(299 rows affected)

(1 row affected)

 Tiempos de ejecución de SQL Server:
   Tiempo de CPU = 0 ms, tiempo transcurrido = 51 ms.

Completion time: 2021-10-15T18:46:41.5643745+02:00
```