

INTRODUCCION

En el ámbito de las bases de datos, crear índices eficaces es primordial para lograr un buen rendimiento de la base de datos, en especial si estamos tratando con grandes volúmenes de información. La ausencia de estos, la sobreindización o el mal diseño de los índices son los principales causantes de problemas de rendimiento de la base de datos.

Un índice en SQL funciona igual que un índice en un libro debido a que provee una forma rápida de localizar información en específica en este mismo. La diferencia es que, en el ámbito de las bases de datos, los índices son una lista ordenada de valores acompañadas de sus punteros que, siendo redundante, apuntan a las paginas de datos donde se encuentran estos valores. Así mismo los propios índices se almacenan en las denominadas paginas de índice.

Un índice es una estructura en disco o en memoria asociada a una tabla o vista que agiliza la recuperación de los registros de la tabla o vista. Un índice contiene claves creadas a partir de los valores de una o varias columnas de la tabla o vista. Almacenan los datos organizados de forma lógica como una tabla con filas y columnas, que a su vez son almacenados físicamente en un formato de datos de fila denominado almacén de filas

En este proyecto estaremos usando los índices agrupados o también conocidos como clustered pero también abordaremos de manera teorica y explicativa los non clustered.

Índice Agrupado (Clustered Index)

Organización física: Este tipo de índice define y almacena el orden físico real de las filas de datos en el disco. Los datos de la tabla se ordenan y almacenan en el disco exactamente en la misma secuencia que el índice.

Cantidad por tabla: Solo puede existir un único índice agrupado por tabla, ya que es imposible que los datos estén físicamente ordenados de más de una manera a la vez.

Analogía: Es como ordenar un archivo de documentos por fecha de manera cronológica; los papeles mismos están físicamente en ese orden.

Índice No Agrupado (Non-Clustered Index)

Organización lógica: Un índice no agrupado no altera el orden físico de los datos en la tabla. En su lugar, crea una estructura de datos independiente y separada de la tabla principal.

Composición: Esta estructura contiene una copia de las columnas indexadas (la clave del índice) junto con punteros o referencias que indican la ubicación física de cada fila de datos correspondiente en la tabla.

Cantidad por tabla: Puede haber múltiples índices no agrupados en una misma tabla.

Analogía: Funciona exactamente como el índice alfabético al final de un libro. El índice te dirige rápidamente a los números de página (los punteros) donde se encuentra la información, sin necesidad de que las páginas del libro estén reordenadas.

Metodologia

Para la realización de este trabajo estaremos usando la siguiente tabla denominada "reserva" que está definida de la siguiente forma:

```
CREATE TABLE reserva
(
    id_reserva INT NOT NULL,
    fecha_reserva DATE NOT NULL,
    cant_personas INT NOT NULL,
    fecha_max_cancelacion DATE NOT NULL,
    id_estado INT NOT NULL,
    id_evento INT NOT NULL,
    dni_cliente INT NOT NULL,
    dni_empleado INT NOT NULL,
    id_rol INT NOT NULL,
    CONSTRAINT pk_reserva PRIMARY KEY (id_reserva),
    CONSTRAINT fk_reserva_estado FOREIGN KEY (id_estado) REFERENCES estado_reserva(id_estado),
    CONSTRAINT fk_reserva_evento FOREIGN KEY (id_evento) REFERENCES evento(id_evento),
    CONSTRAINT fk_reserva_cliente FOREIGN KEY (dni_cliente) REFERENCES cliente(dni_cliente),
    CONSTRAINT fk_reserva_empleado FOREIGN KEY (dni_empleado, id_rol) REFERENCES empleado(dni_empleado, id_rol)
),
```

Para la inserción de un millón de registros en dicha tabla reserva, se usara el siguiente script:

```
SET NOCOUNT ON;

DECLARE @CountClientes INT;
DECLARE @CountEmpleadosRol3 INT;

SELECT @CountClientes = COUNT(*) FROM dbo.cliente;

SELECT @CountEmpleadosRol3 = COUNT(*) FROM dbo.empleado WHERE id_rol = 3;

IF @CountClientes = 0 OR @CountEmpleadosRol3 = 0
BEGIN
    RAISERROR('Error: No se pueden generar reservas sin clientes o sin empleados con ID_ROL = 3.', 16, 1);
    RETURN;
END;

;WITH

L0 AS (SELECT c FROM (VALUES(1),(1)) AS D(c)),
L1 AS (SELECT 1 AS c FROM L0 AS A CROSS JOIN L0 AS B),
L2 AS (SELECT 1 AS c FROM L1 AS A CROSS JOIN L1 AS B),
L3 AS (SELECT 1 AS c FROM L2 AS A CROSS JOIN L2 AS B),
L4 AS (SELECT 1 AS c FROM L3 AS A CROSS JOIN L3 AS B),
L5 AS (SELECT 1 AS c FROM L4 AS A CROSS JOIN L4 AS B),
Generador AS (
    SELECT TOP 1000000
        ROW_NUMBER() OVER(ORDER BY (SELECT NULL)) AS id
    FROM L5
),
ClientesNumerados AS (
    SELECT
        dni_cliente,
        ROW_NUMBER() OVER (ORDER BY NEWID()) AS rn
    FROM dbo.cliente
),
```

```

EmpleadosNumerados AS (
    SELECT
        dni_empleado,
        id_rol,
        ROW_NUMBER() OVER (ORDER BY NEWID()) AS rn
    FROM dbo.empleado
    WHERE id_rol = 3 -- Filtro requerido por la regla de negocio
)
INSERT INTO dbo.reserva (
    fecha_reserva,
    cant_personas,
    id_estado,
    id_evento,
    dni_cliente,
    dni_empleado,
    id_rol
)
SELECT
    DATEADD(second, (ABS(CHECKSUM(NEWID())) % 86400),
    DATEADD(day, - (ABS(CHECKSUM(NEWID())) % 1825), GETDATE())) AS fecha_reserva,

    (ABS(CHECKSUM(NEWID())) % 10) + 1 AS cant_personas,

    1 AS id_estado,

    (ABS(CHECKSUM(NEWID())) % 6) + 1 AS id_evento,

    cli.dni_cliente,

    emp.dni_empleado,
    emp.id_rol -- Siempre será 3

FROM Generador AS g
JOIN ClientesNumerados AS cli
    ON (g.id % @CountClientes) + 1 = cli.rn
JOIN EmpleadosNumerados AS emp
    ON (g.id % @CountEmpleadosRol3) + 1 = emp.rn;
SET NOCOUNT OFF;
PRINT 'Inserción de 1,000,000 de registros completada.';

```

Se eliminan la clave primaria de la tabla reserva y sus relaciones con otras tablas:

```

ALTER TABLE pagos
DROP CONSTRAINT fk_pago_reserva
ALTER TABLE reserva_mesa
DROP CONSTRAINT fk_reserva
ALTER TABLE reserva
DROP CONSTRAINT PK_reserva

```

Se verifica que el índice esté creado correctamente en la tabla reserva:

```

EXECUTE sp_helpindex 'reserva'

```

Script de consulta 1

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'
```

Script de consulta 2

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'  
AND cant_personas BETWEEN '2' AND '9'
```

Script de consulta 3

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'  
AND cant_personas BETWEEN '2' AND '9'  
AND dni_cliente BETWEEN '11000000' AND '15000000'
```

Creamos un índice agrupado sobre la columna fecha_reserva:

```
CREATE CLUSTERED INDEX IX_fecha_reserva ON reserva(fecha_reserva);
```

Se elimina creado el índice en la tabla reserva:

```
DROP INDEX IX_fecha_reserva ON reserva;
```

Creamos un índice agrupado sobre la columna fecha_reserva incluyendo otras columnas en base a las cuales se realizan las consultas:

```
CREATE CLUSTERED INDEX IX_fecha_reserva ON reserva(fecha_reserva, cant_personas, dni_cliente);
```

Muestra estadísticas detalladas sobre el tiempo de ejecución:

```
SET STATISTICS TIME ON
```

Muestra estadísticas de operaciones de E/S (lectura/escritura):

```
SET STATISTICS IO ON  
|
```

Desarrollo

Todos los script y el desarrollo del trabajo se realizara en Microsoft SQL Server 2021, utilizando de la herramienta de Plan de Ejecución Actual de dicho programa

Sobre la tabla elegida se ejecutara el script para cargar un millón de registros y posteriormente se ejecutara el siguiente script para eliminar la Primary Key y sus relaciones con otras tablas:

```
ALTER TABLE pagos
DROP CONSTRAINT fk_pago_reserva
ALTER TABLE reserva_mesa
DROP CONSTRAINT fk_reserva
ALTER TABLE reserva
DROP CONSTRAINT PK_reserva
```

Se ejecutara el siguiente Script para confirmar dicha eliminación:

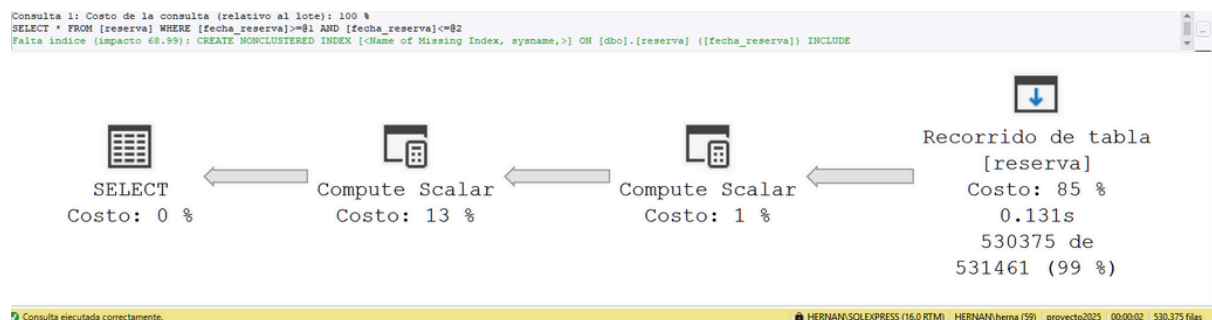
```
EXECUTE sp_helpindex 'reserva'
```

Luego de procedera a empezar con las pruebas de los Scripts de Consulta:

```
SELECT *
FROM reserva
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'
```

Tiempos de Ejecucion:

- Primer Intento 2062ms/2,062seg
- Segundo Intento 2051ms/2,051seg
- Tercer Intento 2089ms/2,089seg

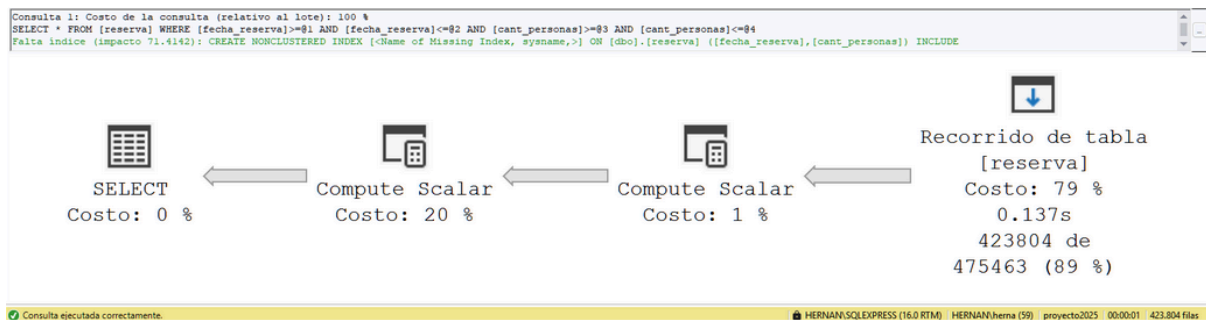


Script 2:

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'  
AND cant_personas BETWEEN '2' AND '9'
```

Tiempos de Ejecucion:

- Primer Intento 1706 ms/1,706 seg
- Segundo Intento 1715 ms/1,715 seg
- Tercer Intento 1558 ms/1,558 seg

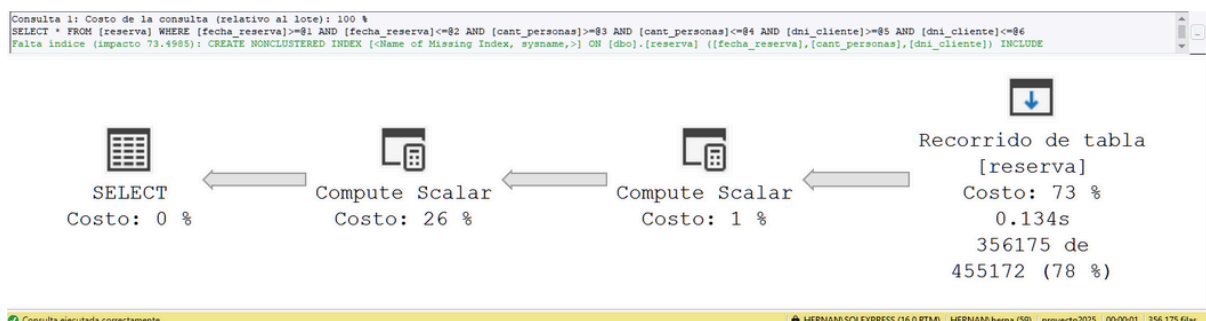


Script 3:

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'  
AND cant_personas BETWEEN '2' AND '9'  
AND dni_cliente BETWEEN '11000000' AND '15000000'
```

Tiempos de Ejecucion:

- Primer Intento 1435 ms/1,435 seg
- Segundo Intento 1501 ms/1,501 seg
- Tercer Intento 1551 ms/1,551 seg



Luego se creara un Indice Clustered sobre la columna fecha_reserva con el siguiente comando:

```
CREATE CLUSTERED INDEX IX_fecha_reserva ON reserva(fecha_reserva);
```

Se ejecutara nuevamente este comando para confirmar que dicho índice esta correctamente creado:

```
EXECUTE sp_helpindex 'reserva'
```

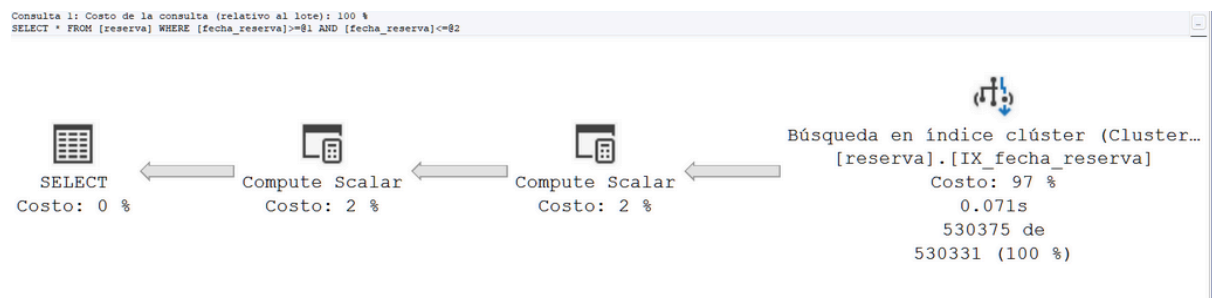
Se procede a ejecutar las consultas previamente dichas con los siguientes resultados:

Script 1 con índice:

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'
```

Tiempos de Ejecucion:

- Primer Intento 1974 ms/1,974 seg
- Segundo Intento 2006 ms/2,006 seg
- Tercer Intento 2001 ms/2,001 seg

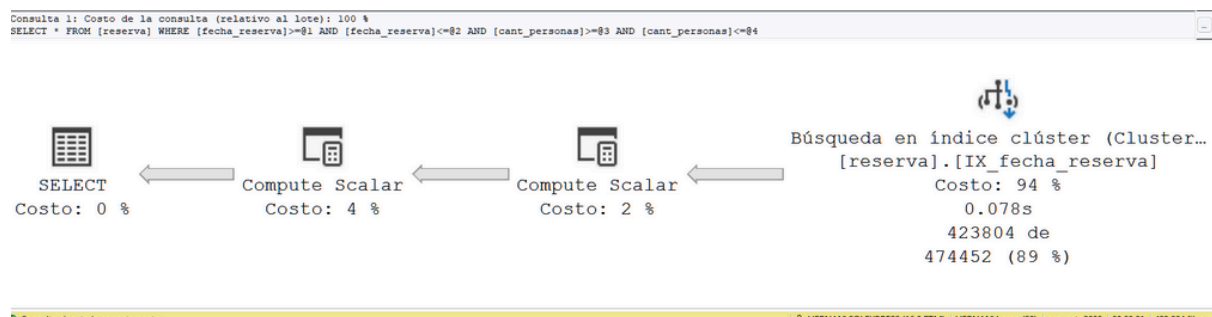


Script 2 con índice:

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'  
AND cant_personas BETWEEN '2' AND '9'
```

Tiempos de Ejecucion:

- Primer Intento 1711 ms ms/1,711 seg
- Segundo Intento 1669 ms/1,669 seg
- Tercer Intento 1668 ms/1,668 seg

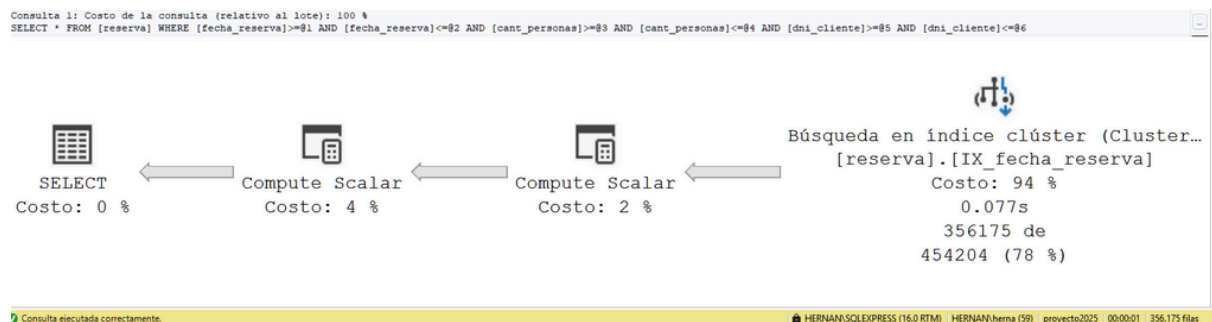


Script 3 con índice:

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'  
AND cant_personas BETWEEN '2' AND '9'  
AND dni_cliente BETWEEN '11000000' AND '15000000'
```

Tiempos de Ejecucion:

- Primer Intento 1420 ms./1,420 seg
- Segundo Intento 1400 ms/1,400 seg
- Tercer Intento 1398 ms/1,398 seg



Se elimina nuevamente el índice en la tabla reserva:

```
DROP INDEX IX_fecha_reserva ON reserva;
```

Se verifica que el índice se haya eliminado correctamente:

```
EXECUTE sp_helpindex 'reserva'
```

Creamos un índice agrupado sobre la columna fecha_reserva incluyendo otras columnas en base a las cuales se realizan las consultas:

```
CREATE CLUSTERED INDEX IX_fecha_reserva ON reserva(fecha_reserva, cant_personas, dni_cliente);
```

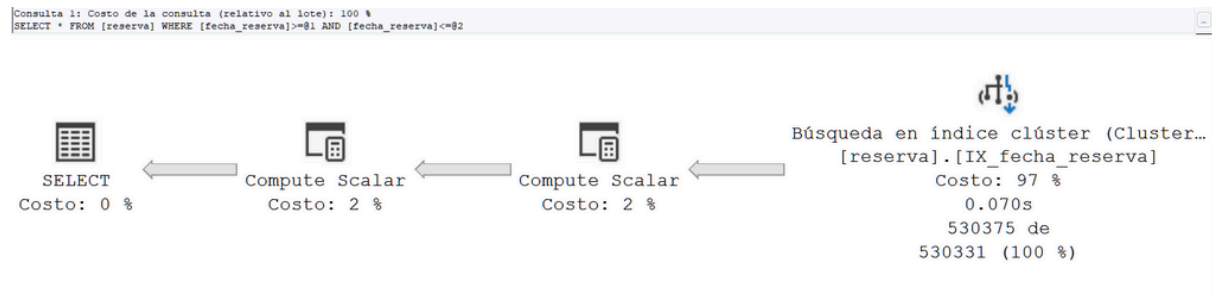
Se ejecutan nuevamente las consultas por ultima vez con los siguientes resultados:

Script 1 con índice compuesto:

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'
```

Tiempos de Ejecucion:

- Primer Intento 2009 ms/2,009 seg
- Segundo Intento 2010 ms/2,010 seg
- Tercer Intento 2018 ms/2,018 seg

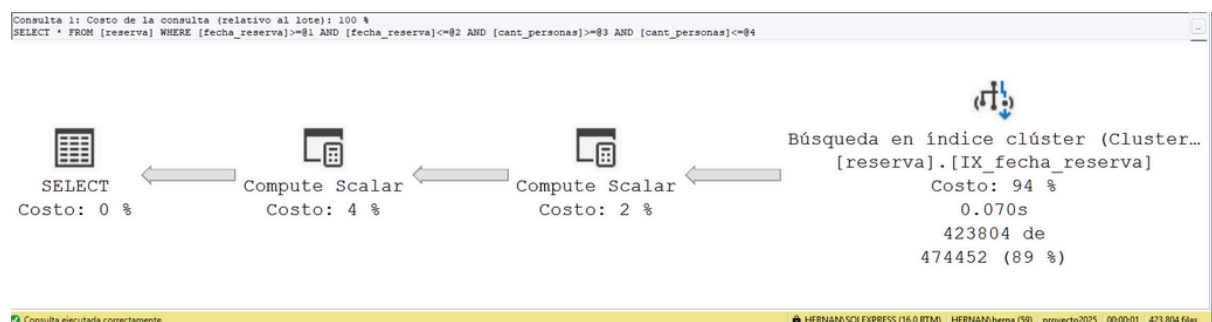


Script 2 con índice compuesto:

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'  
AND cant_personas BETWEEN '2' AND '9'
```

Tiempos de Ejecucion:

- Primer Intento 1634 ms/1,634 seg
- Segundo Intento 1647 ms/1,647 seg
- Tercer Intento 1656 ms./1,656 seg

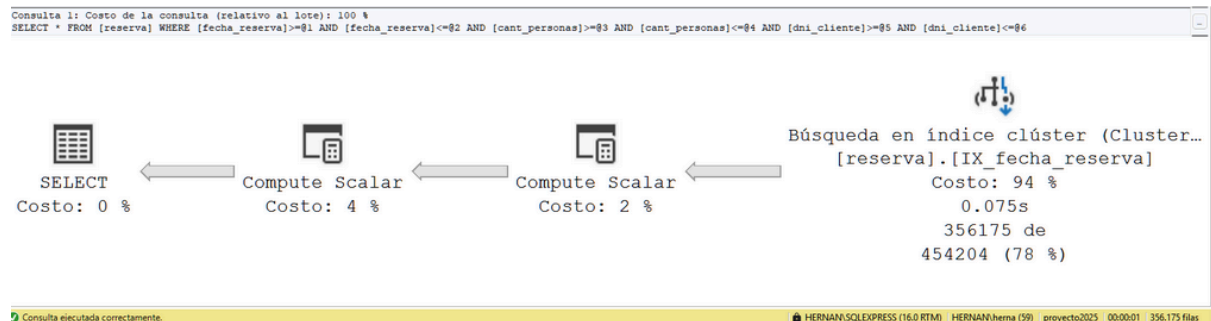


Script 3 con índice compuesto:

```
SELECT *  
FROM reserva  
WHERE fecha_reserva BETWEEN '2022-03-14' AND '2024-11-07'  
AND cant_personas BETWEEN '2' AND '9'  
AND dni_cliente BETWEEN '11000000' AND '15000000'
```

Tiempos de Ejecucion:

- Primer Intento 1418 ms./1,418 seg
- Segundo Intento 1385 ms/1,385 seg
- Tercer Intento 1406 ms/1,406 seg



Conclusión

Luego de realizar todas las pruebas anteriormente dichas, podemos observar una mejora en el tiempo de respuesta de las consultas sobre la tabla reserva al aplicar los índices en la columna fecha. Gracias a la herramienta de Plan de Ejecución de SQL Server, podemos ver una mejora de casi el 50% en las consultas cuando la tabla posee índice respecto a cuando no lo posee; esta mejora, que en este caso parece mínima, crecerá exponencialmente de acuerdo al flujo de datos con el que se trabaje. Por ende, podemos decir que los índices y su buena aplicación son fundamentales para el buen funcionamiento de toda Base de Datos y la optimización de las consultas sobre esta misma.

Bibliografía

-Microsoft(1 de Octubre de 2025). Guía de diseño y arquitectura de índices.Microsoft Learn.
Recuperado el 10 de Noviembre de 2025 desde:

https://learn.microsoft.com/es-es/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver17#hash_index

-Excel y Mas(2015, 28 de Junio). Creación de Índices | Curso de SQL Server #12[Video].
Youtube. <https://www.youtube.com/watch?v=y1TxR53RIYU>

-Greg Robidoux(5 de Junio de 2025).Index Scans and Table Scans. MSSQLTips.
Recuperado el 11 de Noviembre de 2025 desde:

<https://www.mssqltips.com/tutorial/index-scans-and-table-scans/>