Licenciatura en Sistemas de Información

Bases de Datos I

Tema de Investigación – Vistas y Vistas Indexadas

Grupo n° 17:

Asselborn, Santiago

Gerber, Federico

Larroza, Lautaro

Laola, Mariano

Año: 2.025

PARTE TEÓRICA

1. Concepto de Vistas en SQL

Una vista es una tabla virtual que se genera a partir de una consulta SQL sobre una o más tablas reales. Su función principal es simplificar el acceso a los datos, restringir columnas o filas visibles y facilitar la reutilización de consultas complejas. A diferencia de una tabla, una vista no almacena físicamente los datos, sino que los obtiene dinámicamente al ser consultada.

Las ventajas que podemos obtener del uso de las vistas son los siguientes:

- Simplicidad: nos permite reducir la complejidad de consultas frecuentes.
- Seguridad: nos permite ocultar información sensible de una tabla en específico.
- Reutilización: nos facilita la creación de informes o consultas estándar.
- Independencia lógica: en el caso de que cambie la estructura de la tabla, se puede ajustar la vista sin afectar las consultas de los usuarios.

2. Vistas Actualizables

Una vista es actualizable cuando permite realizar operaciones de inserción, actualización o eliminación (CRUD) que impacten directamente sobre las tablas base. Para que una vista sea actualizable, debe cumplir ciertas condiciones:

- Basarse en una única tabla.
- No incluir funciones de agregación, DISTINCT, GROUP BY, UNION, ni sub consultas complejas.
- Incluir todas las columnas requeridas por restricciones NOT NULL o PRIMARY KEY de la tabla base.

3. Vistas Indexadas

Una vista indexada es una vista materializada: los datos resultantes de su consulta se almacenan físicamente junto con un índice clausterizado (agrupado o categorizado). Esto mejora significativamente el rendimiento de las consultas frecuentes o agregaciones pesadas, a cambio de un mayor costo de almacenamiento y mantenimiento.

Los requisitos para crearla son los siguientes:

- La vista debe crearse con la opción WITH SCHEMABINDING.
- Debe contener solo columnas deterministas y referenciar tablas de un mismo esquema.
- No puede contener TOP, DISTINCT, TEXT /NTEXT/IMAGE, ni funciones no deterministas.
- El primer índice creado sobre la vista debe ser clausterizado y único.

Las ventajas de una vista indexada son las siguientes:

- Mejora en el rendimiento de reportes o consultas agregadas.
- Permite mantener datos previamente calculados.
- Reduce el tiempo de ejecución en análisis frecuentes.

A su vez, también podemos observar algunas desventajas:

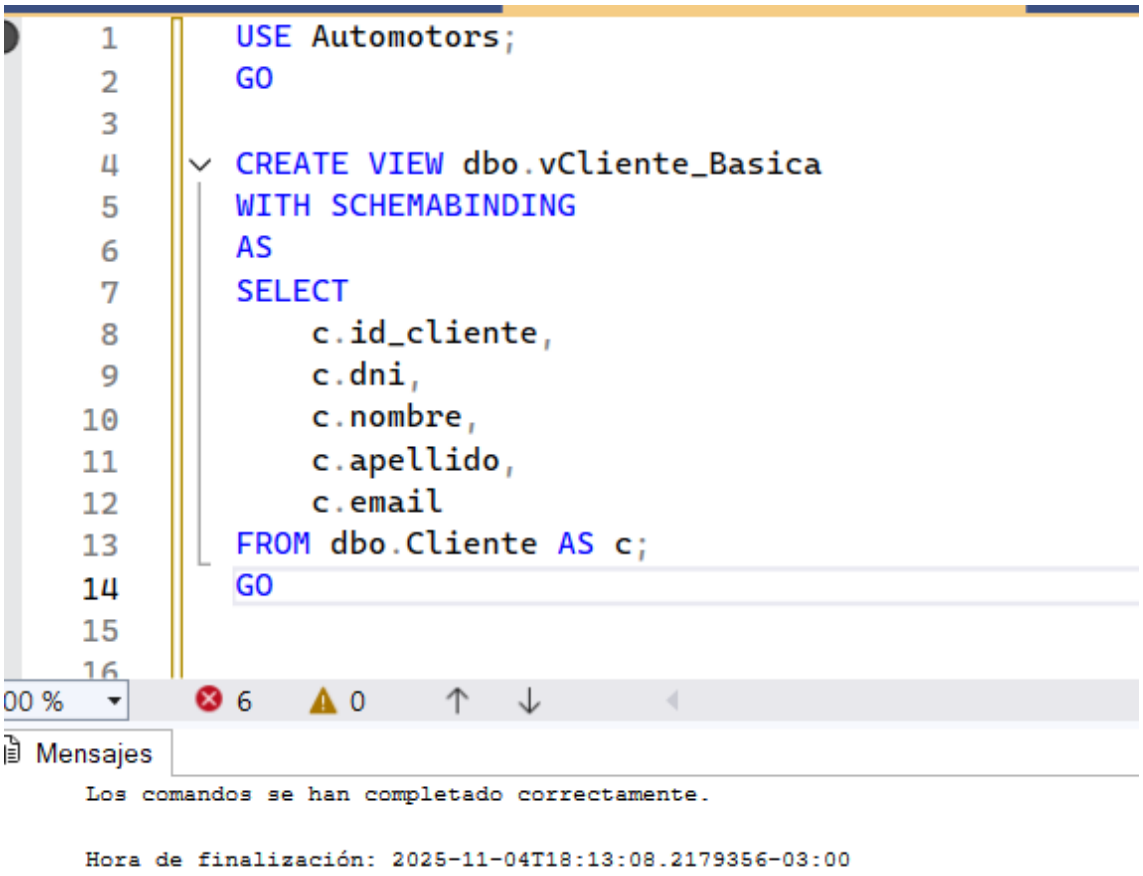
- Requiere espacio adicional en disco.
- Aumenta el costo de mantenimiento en operaciones INSERT, UPDATE o DELETE.

PARTE PRÁCTICA

Luego de esta breve introducción, realizamos las pruebas pertinentes sobre la base de datos “Automotors”, aplicando los conceptos adquiridos.

1. Creación de una vista simple

Primero, creamos una vista de la tabla “Cliente”:



Hacemos uso de WITH SCHEMABINDING para garantizar que la vista quede vinculada de forma permanente al esquema de la tabla Cliente, impidiendo que se modifique su estructura mientras la vista exista.

2. Inserción de datos a través de la vista

Una vez creada la vista sobre la tabla “Cliente”, podemos realizar la inserción de 3 nuevos clientes:

```sql
USE Automotors;
GO

INSERT INTO dbo.vCliente_Basica (dni, nombre, apellido, email)
VALUES
('44573877','Santiago','Asselborn','santiasselborn1@gmail.com'),
('10000000','Bruno','López','brunolopez@gmail.com'),
('11000000','Carla','Gómez',NULL);

SELECT id_cliente, dni, nombre, apellido, email
FROM dbo.Cliente
WHERE dni IN ('44573877','10000000','11000000');
```

100 % 5 0

Resultados Mensajes

	id_cliente	dni	nombre	apellido	email
1	1	44573877	Santiago	Asselborn	santiasselborn1@gmail.com
2	2	10000000	Bruno	López	brunolopez@gmail.com
3	3	11000000	Carla	Gómez	NULL

3. Actualización de registros mediante la vista

Actualizamos el correo del cliente Carla Gómez:

```sql
USE Automotors;
GO

UPDATE v
SET email = 'carlagomez@gmail.com'
FROM dbo.vCliente_Basica AS v
WHERE v.dni = '11000000';

SELECT id_cliente, dni, nombre, apellido, email
FROM dbo.Cliente
WHERE dni = '11000000';
```

100 % 4 0

Resultados Mensajes

	id_cliente	dni	nombre	apellido	email
1	3	11000000	Carla	Gómez	carlagomez@gmail.com

4. Eliminación de registros a través de la vista

```
1  USE Automotors;
2  GO
3
4  DELETE v
5  FROM dbo.vCliente_Basica AS v
6  WHERE v.dni IN ('44573877','10000000','11000000');
7
8  SELECT id_cliente, dni, nombre, apellido, email
9  FROM dbo.Cliente
10 WHERE dni IN ('44573877','10000000','11000000');
11
12
13
14
15
16
```

100 % 3 0

Resultados | Mensajes

id_cliente	dni	nombre	apellido	email

5. Creación de vista indexada

Ahora creamos una vista indexada sobre la vista Cliente creada anteriormente:

```
1  USE Automotors;
2  GO
3
4  CREATE UNIQUE CLUSTERED INDEX IXC_vCliente_Basica_dni
5      ON dbo.vCliente_Basica(dni);
6  GO
7
8  CREATE NONCLUSTERED INDEX IX_vCliente_Basica_apellido
9      ON dbo.vCliente_Basica(apellido, nombre);
10 GO
11
12
13
14
15
16
```

0 % 10 0

Mensajes

```
Los comandos se han completado correctamente.

Hora de finalización: 2025-11-04T18:26:58.5904440-03:00
```

6. Evaluación de rendimiento en tiempo real:

```
1   USE Automotors;
2   GO
3
4   SET STATISTICS IO ON;
5   SET STATISTICS TIME ON;
6   GO
7
8   SELECT id_cliente, nombre, apellido, email
9   FROM dbo.vCliente_Basica
10  WHERE dni = '44573877';
11  GO
12
13  SELECT id_cliente, nombre, apellido, email
14  FROM dbo.Cliente
15  WHERE dni = '44573877';
16  GO
17
18  SELECT id_cliente, dni, nombre, apellido
19  FROM dbo.vCliente_Basica
20  WHERE apellido = 'Asselborn';
21  GO
22
23  SET STATISTICS IO OFF;
24  SET STATISTICS TIME OFF;
25  GO
26
27
```

100 % 13 0

Resultados Mensajes

	id_cliente	nombre	apellido	email
1	4	Santiago	Asselborn	santiasselborn@gmail.com

	id_cliente	nombre	apellido	email
1	4	Santiago	Asselborn	santiasselborn@gmail.com

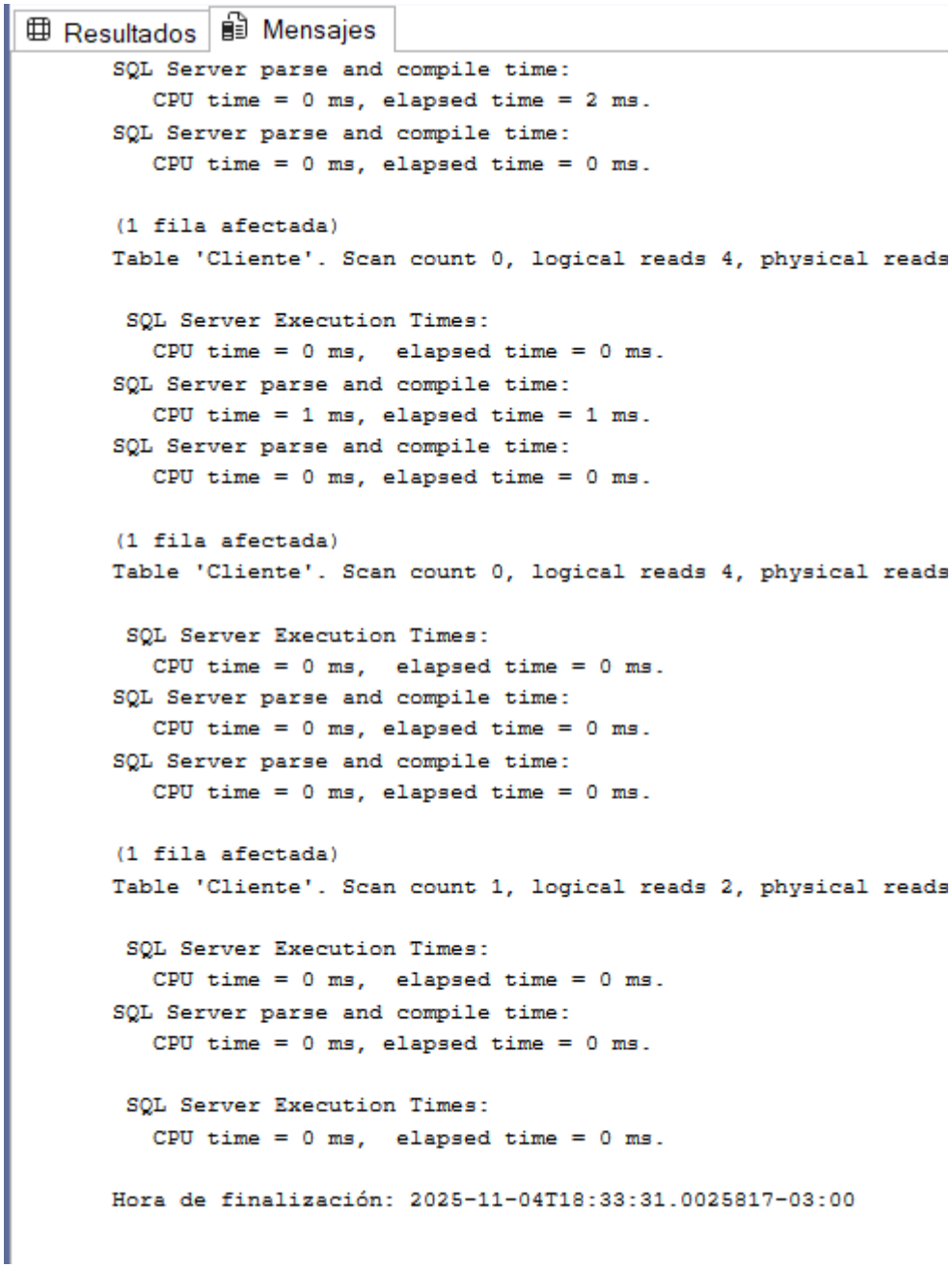
	id_cliente	dni	nombre	apellido
1	4	44573877	Santiago	Asselborn

- Primera ejecución:

Resultados | Mensajes

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 2 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(1 fila afectada)
Table 'Cliente'. Scan count 0, logical reads 4, physical reads

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 7 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(1 fila afectada)
Table 'Cliente'. Scan count 0, logical reads 4, physical reads

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(1 fila afectada)
Table 'Cliente'. Scan count 1, logical reads 2, physical reads

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Hora de finalización: 2025-11-04T18:30:16.8556403-03:00
```

- Segunda ejecución:



7. Conclusiones

A partir de las pruebas realizadas con vistas y vistas indexadas en la base de datos Automotors, se pudo comprobar que estas estructuras representan una herramienta valiosa tanto para la organización lógica de la información como para la optimización del rendimiento de las consultas.

En primer lugar, la creación de una vista simple sobre la tabla "**Cliente**" permitió restringir la visualización únicamente a ciertos campos relevantes (como lo son: dni, nombre, apellido, email), facilitando el acceso controlado a los datos sin exponer toda la estructura original.

Asimismo, se verificó que las operaciones CRUD (inserción, actualización y eliminación) realizadas a través de la vista se reflejaron correctamente en la tabla base, confirmando que este tipo de vistas son totalmente actualizables.

En segundo lugar, la implementación de una vista indexada con un índice clustered (índice agrupado, encargado del orden real dentro del disco) único sobre el campo **dni** permitió observar una mejora notable en el rendimiento de las consultas repetitivas.

Mediante el uso de las opciones **SET STATISTICS IO** y **SET STATISTICS TIME**, se constató que las lecturas lógicas y el tiempo de ejecución disminuyeron al consultar la vista indexada respecto de la tabla original (evidenciando que se puede aprovechar el índice de la vista para acceder de forma más eficiente a los datos, reduciendo el costo de acceso y acelerando las búsquedas).

Pese a ello, también se observó que este tipo de optimización implica un mayor costo de mantenimiento: cada vez que se inserta, modifica o elimina un registro en la tabla base, el índice asociado a la vista debe actualizarse, lo cual puede impactar levemente en el rendimiento de las operaciones de escritura.

En conclusión, las vistas simples son ideales para simplificar y controlar el acceso a la información, mientras que las indexadas resultan especialmente útiles en escenarios donde se realizan consultas frecuentes y predecibles sobre los mismos campos. En el caso de Automotors, una vista indexada sobre los datos de clientes mejora la eficiencia en las búsquedas por DNI o apellido.