Universiada Nacional del Nordeste

VISTAS Y VISTAS INDEXADAS

Profesor: Dario Villegas
Integrantes: Matias, Freixes
Mauricio, Fernando
Alejandro, Jimenez

Licenciatura en Sistemas.
OCTUBRE /2023

INDICE

CAPITULO I: INTRODUCCIÓN....3
CAPITULO II: MARCO CONCEPTUAL....4
CAPITULO III: METODOLOGÍA SEGUIDA....5
CAPITULO IV: DESARROLLO DEL TEMA....6
VISTAS....6
Utilidades de las vistas....6
Crear vistas (CREATE) - Requisitos adicionales....7
Uso de Transact-SQL....7
Modificar vistas (ALTER) - Requisitos adicionales....8
Uso de Transact-SQL....8
Actualizar vistas (UPDATE) - Requisitos adicionales....9
Uso de Transact-SQL....9
Eliminar vistas (DROP) - Requisitos adicionales....10
Uso de Transact-SQL....10
VISTAS INDEXADAS....11
Crear índices en una vista....11
Pasos para la generación de Vistas Indexadas....12
Cláusulas para indexación de vistas....14
Requisitos adicionales....15
Diferencia entre Vistas y Vistas indexadas:....16
Cómo agregar permisos a una vista....17
CAPITULO V: CONCLUSIONES....19
CAPITULO VI: BIBLIOGRAFÍA....20

CAPITULO I: INTRODUCCIÓN

En el ámbito de la gestión de bases de datos, el uso eficiente de SQL es esencial. Este trabajo práctico se centra en un aspecto clave: "SQL - Vistas y Vistas Indexadas". Estas herramientas ofrecen soluciones a desafíos comunes en la organización y recuperación de datos en bases de datos relacionales.

El problema que abordaremos se relaciona con la comprensión y aplicación efectiva de vistas y vistas indexadas en SQL, y cómo pueden mejorar la eficiencia de las consultas y simplificar la gestión de datos.

Nuestro objetivo general es explorar y explicar el concepto y la utilidad de las vistas y vistas indexadas. Además, buscamos analizar cómo su implementación beneficia el rendimiento de las consultas SQL y optimiza las bases de datos relacionales.

Los objetivos específicos incluyen:

- Identificar situaciones adecuadas para el uso de vistas y vistas indexadas.
- Presentar ejemplos concretos de su implementación.
- Evaluar el impacto en la eficiencia de las consultas con y sin su uso.

CAPITULO II: MARCO CONCEPTUAL

En este trabajo, para aprender vistas y vistas indexadas en SQL Server, es esencial comprender el uso de consultas a tablas y cláusulas internas para el manejo de seguridad y rendimiento de consultas.
Adicional, nuestro campo de trabajo será a través de una base de datos Demo de un consorcio, el cual suministramos a continuación el diagrama de relación.

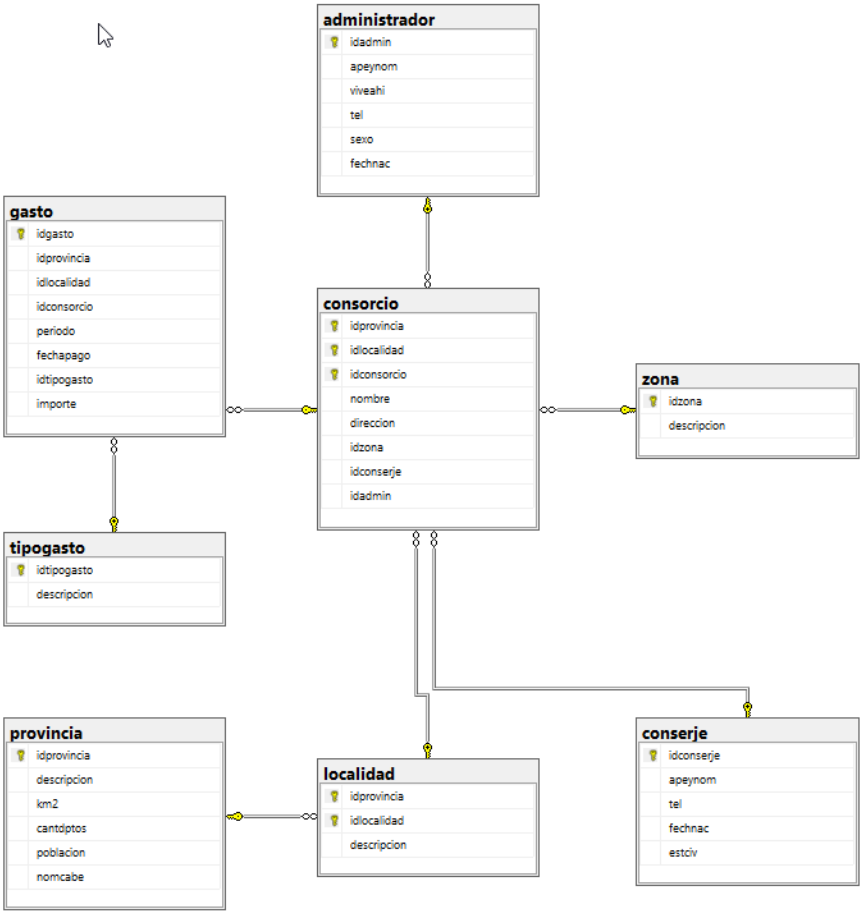


Diagrama de relación de la DB base_consorcio.

También hemos agregado un total de 1080000 (un millón ochenta mil) inserciones en la tabla de Gasto, de esta misma realizaremos las consultas en las próximas vistas y permitir conocer la estructura en que se maneja la vista indexada.

CAPITULO III: METODOLOGÍA SEGUIDA

Para profundizar en el tema de las vistas y vistas indexadas en SQL, iniciamos nuestra investigación mediante la búsqueda de información en línea. Exploramos una variedad de recursos, incluyendo sitios web, blogs especializados y documentación técnica. Esto nos permitió obtener una perspectiva integral de las vistas en SQL, sus ventajas y desventajas, así como la importancia de las vistas indexadas en la optimización del rendimiento de consultas.
La comunicación efectiva fue esencial para nuestro proyecto de investigación. Utilizamos plataformas de mensajería como WhatsApp y Discord para mantener una comunicación constante entre los miembros del equipo. Estas aplicaciones nos permitieron intercambiar ideas, resolver dudas y coordinar nuestras tareas de manera eficiente, a pesar de estar con tiempos discontinuos a la hora de organizarnos.

Para la creación de nuestro documento escrito, optamos por Google Drive y su herramienta Docs. Esta elección nos brindó la ventaja de contar con un documento accesible en línea, lo que facilitó la colaboración en tiempo real y la revisión conjunta de contenido. Además, Google Docs nos permitió llevar un registro de las ediciones y comentarios de cada miembro del equipo, lo que resultó fundamental para mantener un seguimiento preciso de los cambios realizados.

Para la presentación del proyecto, utilizamos Canvas para diseñar una portada atractiva y profesional. Canvas nos permite personalizar la portada con gráficos e imágenes, lo que contribuyó a darle un aspecto visualmente agradable y atractivo.

Para la realización de scripts de prueba relacionados con vistas y vistas indexadas, utilizamos Microsoft SQL Server Management Studio. Esta herramienta nos permitió escribir y ejecutar scripts SQL de manera eficiente, lo que fue crucial para comprender en la práctica los conceptos que estábamos investigando. Además, durante todo el proceso de investigación y elaboración de nuestro proyecto, utilizamos GitHub como un sistema de control de versiones. Esto nos permitió rastrear y documentar los cambios realizados en los scripts creados para la prueba y carga de datos en la base de datos.

En resumen, nuestra investigación sobre "Vistas y Vistas Indexadas en SQL" fue un proceso colaborativo que involucró la búsqueda de información en línea, la comunicación constante a través de aplicaciones de mensajería y la creación de un documento accesible en línea mediante Google Drive, complementado con una portada diseñada en Canvas. Estas herramientas y enfoques nos permitieron llevar a cabo el proyecto esperando que se encuentre con las normas e indicaciones necesarias para su valoración.

CAPITULO IV: DESARROLLO DEL TEMA

VISTAS

Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla, una vista consta de un conjunto de columnas y filas de datos con un nombre. Sin embargo, a menos que esté indexada, una vista no existe como conjunto de valores de datos almacenados en una base de datos. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista.

Las VISTAS, se utilizan a menudo como filtros de las tablas de las que recogen datos y es por ello por lo que se utilizan como mecanismos de seguridad, ya que permiten al usuario tener acceso de visibilidad de los datos, pero no a las tablas reales subyacentes. También se utilizan para simplificar y personalizar la visualización de los datos para el usuario.

En SQL Server, existen diferentes tipos de vistas:

- **Vistas de usuario**: tablas virtuales definidas por el usuario cuyo contenido está definido por una consulta.
- **Vistas indexadas**: cuyos datos se han almacenado como una tabla real ya que se han indizado a través de un índice clúster único. Este tipo de vistas mejoran mucho el rendimiento en algunos tipos de consultas que devuelven muchos registros.
- **Vistas con particiones**: combinan datos horizontales con particiones de un conjunto de tablas miembro en uno o más servidores.
- **Vistas del sistema**: específicas para consultar metadatos del sistema.

Utilidades de las vistas

Puede crear vistas en el motor de base de datos de SQL Server mediante SQL Server Management Studio o Transact-SQL. Se puede usar una vista para lo siguiente:

- Permiten centrar, y alterar la percepción de la base de datos para cada usuario
- Un mecanismo de seguridad ya que genera una suerte de virtualización/acceso directo de la tabla y solo permite ver cierta cantidad de datos de esa tabla, pero no se les permite ver las tablas subyacentes a esa vista
- Proporciona una interfaz compatible con versiones anteriores para emular una tabla cuyo esquema podría haber cambiado

Utilizaremos el esquema de base de un consorcio, el cual detallamos a continuación el diagrama de relación que es necesario conocer para entender de donde se toman los datos para la generación de vistas y vistas indexadas.

Crear vistas (CREATE) - Requisitos adicionales

Limitaciones y restricciones	Una vista solo se puede crear en la base de datos actual.
	Una vista puede tener un máximo de 1.024 columnas.
Permisos	Se necesita el permiso `CREATE VIEW` en la base de datos y el permiso `ALTER` en el esquema en que se crea la vista.

Tabla de requisitos adicionales para crear vistas.

Uso de Transact-SQL

1. En el Explorador de objetos, conéctese a una instancia del Motor de base de datos.
2. En la barra Estándar, seleccione Nueva consulta.
3. Escriba la sentencia `CREATE VIEW` seguido del nombre para la vista, luego proceda a escribir la consulta.
4. Procesa a realizar un `SELECT * FROM` a la vista y observe los resultados.

```
2  ----------------------------------------
3  -- CREAR VIEW administrador_view
4  ----------------------------------------
5  CREATE VIEW administrador_view AS
6  (
7      SELECT
8      apeynom,
9      sexo,
10     fechnac
11     FROM [dbo].[administrador]
12 )
13  GO
14 ----------------------------------------
15 -- Ver los datos en la view
16 ----------------------------------------
17 SELECT * FROM [dbo].[administrador_view]
```

Crear view T-SQL.

Modificar vistas (ALTER) - Requisitos adicionales

Limitaciones y restricciones
● La modificación de una vista no afecta a ningún objeto dependiente, como procedimientos almacenados o desencadenadores, a menos que la definición de la vista cambie de forma que el objeto dependiente ya no sea válido. ● Si una vista que está actualmente en uso se modifica mediante `ALTER VIEW`, el Motor de base de datos impone un bloqueo exclusivo de esquema sobre la vista. Cuando se concede el bloqueo, y no hay usuarios activos de la vista, el Motor de base de datos elimina todas las copias de la vista de la caché de procedimientos. Los planes existentes que hacen referencia a la vista permanecen en la caché, pero se vuelven a compilar cuando se llaman. ● `ALTER VIEW` se puede aplicar a vistas indexadas; no obstante, quita incondicionalmente todos los índices de la vista. ● Para ejecutar `ALTER VIEW`, como mínimo, se necesita el permiso `ALTER` en `OBJECT`.

Tabla de requisitos adicionales para modificar vistas.

Uso de Transact-SQL

1. En el Explorador de objetos, conéctese a una instancia del Motor de base de datos.
2. En la barra Estándar, seleccione Nueva consulta.
3. Siga los pasos de la imagen donde se muestra la consulta y seleccione Ejecutar. El ejemplo crea primero una vista y luego la modifica mediante `ALTER VIEW`. En donde hemos agregado el campo “tel”.
4. En este caso, al realizar el alter seguimos obteniendo los mismos datos anteriores de la tabla administrador, solo que esta vez con un campo adicional.

```sql
USE [base_consorcio];
-------------------------------------
-- CREAR VIEW administrador_view
-------------------------------------
ALTER VIEW [dbo].[administrador_view] AS
(
    SELECT
        apeynom,
        sexo,
        fechnac,
        tel --ESTE ES EL NUEVO CAMPO
    FROM  [dbo].[administrador]
)
GO
-------------------------------------
-- Ver los cambios en la view
-------------------------------------
SELECT * FROM administrador_view
```

Modificar view T-SQL.

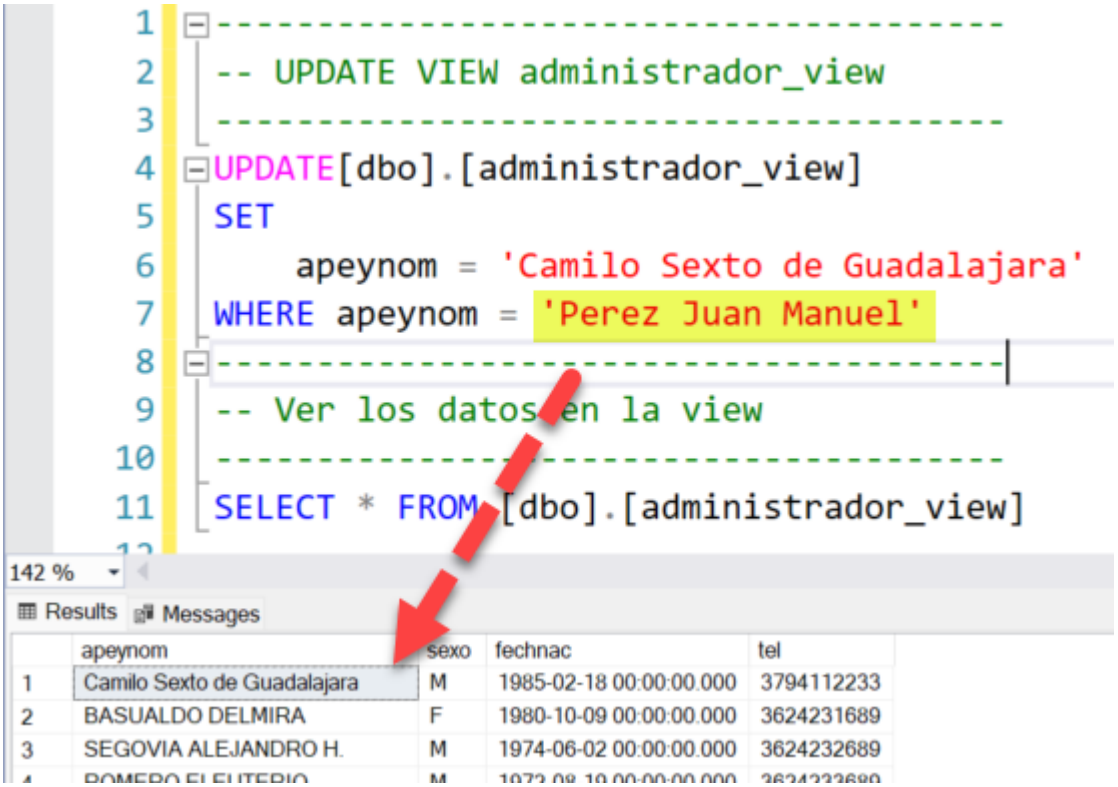
Actualizar vistas (UPDATE) - Requisitos adicionales

Limitaciones y restricciones	Una vista solo se puede actualizar en la base de datos actual.
	Una vista puede tener un máximo de 1.024 columnas.
Permisos	Requiere los permisos `UPDATE`, `INSERT` o `DELETE` en la tabla de destino, en función de la acción que se realizará.

Tabla de requisitos adicionales para actualizar vistas.

Uso de Transact-SQL

1. En el Explorador de objetos, conéctese a una instancia del Motor de base de datos.
2. En la barra de Estándar, haga clic en Nueva consulta.
3. Ejecute la acción mostrada en la imagen.
4. Para este caso, hemos utilizado como condición que sólo se modifique el apeynom que posea ‘Perez Juan Manuel’, a través de una vista podemos modificar los datos originales de la tabla base.



Actualizar tabla con view T-SQL.

La instrucción se realiza correctamente porque solo se especifican la columna de una tabla base y las demás columnas de la tabla base tienen valores predeterminados.

Eliminar vistas (DROP) - Requisitos adicionales

Limitaciones y restricciones	Cuando se quita una vista, la definición y otra información de la vista se elimina del catálogo del sistema. También se eliminan todos los permisos de la vista.
	Las vistas de una tabla que se ha quitado mediante `DROP TABLE` se deben quitar explícitamente con `DROP VIEW`.
Permisos	Se necesita el permiso `ALTER` en `SCHEMA` o el permiso `CONTROL` en `OBJECT`.

Tabla de requisitos adicionales para eliminar vistas.

Uso de Transact-SQL

1. En el Explorador de objetos, conéctese a una instancia del Motor de base de datos.
2. En la barra Estándar, seleccione Nueva consulta.

3. Siga los pasos de la imagen donde se muestra la consulta y seleccione Ejecutar. en esta consulta verificamos que efectivamente la vista a eliminar existe registrada, con el fin de evitar errores.

```
1  ----------------------------------------
2  -- ELIMINAR VIEW administrador_view
3  ----------------------------------------
4  IF OBJECT_ID('[dbo].[administrador_view]', 'V') IS NOT NULL
5  DROP VIEW [dbo].[administrador_view]
6
7
8
```

Eliminar vista verificando OBJECT_ID con T-SQL.

También podemos usar la sintaxis `IF EXISTS`, introducida en SQL Server 2016

```
1  ----------------------------------------
2  -- ELIMINAR VIEW administrador_view
3  ----------------------------------------
4  DROP VIEW IF EXISTS[dbo].[administrador_view]
5
6
7
```

Eliminar vista verificando existencia con IF EXISTS T-SQL.

VISTAS INDEXADAS

Una vista indexada es una vista que se ha materializado. Esto significa que se ha calculado la definición de la vista y que los datos resultantes se han almacenado como una tabla. Se puede indicar una vista creando un índice clúster único en ella. Las vistas indexadas pueden mejorar de forma considerable el rendimiento de algunos tipos de consultas. Las vistas indexadas funcionan mejor para consultas que agregan muchas filas. No son adecuadas para conjuntos de datos subyacentes que se actualizan frecuentemente.

Crear índices en una vista

El primer índice creado en una vista debe ser un **índice clúster único**. Después de haber creado el índice clúster único, puede crear más índices no clúster. La creación de un índice clúster único en una vista mejora el rendimiento de las consultas, ya que la vista se almacena en la base de datos de la misma manera que se almacena una tabla con un índice agrupado. El optimizador de consultas puede utilizar vistas indexadas para acelerar

la ejecución de las consultas. No es necesario hacer referencia a la vista en la consulta para que el optimizador tenga en cuenta esa vista para una sustitución.

Pasos para la generación de Vistas Indexadas

Para crear una vista indexada, es necesario seguir los pasos descritos a continuación, que son fundamentales para la correcta implementación de la vista indexada:

1. Compruebe que las opciones SET son correctas para todas las tablas existentes a las que se hará referencia en la vista.
2. Compruebe que las opciones SET de la sesión están establecidas correctamente antes de crear cualquier tabla y la vista.
3. Compruebe que la definición de vista sea determinista.
4. Verifique que la tabla base y la vista tengan el mismo propietario.
5. Cree la vista con la opción WITH SCHEMABINDING.
6. Cree el índice clúster único en la vista.

veámoslo en un ejemplo con Sql Server:

1. En el Explorador de objetos, conéctese a una instancia del Motor de base de datos.
2. En la barra Estándar, seleccione Nueva consulta.
3. En la siguiente consulta, crearemos una vista indexada llamada `ConsorcioDetalles_View` , tiene en su esquema de consulta relaciones con otras tablas a través del uso de `INNER JOIN`.

```sql
USE [base_consorcio]
GO
-------------------------------------
-- CREAR VIEW ConsorcioDetalles_view
-------------------------------------
CREATE OR ALTER VIEW [dbo].[ConsorcioDetalles_view]
WITH SCHEMABINDING
AS
(

    SELECT
        IdGasto         = G.idgasto,
        Administrador   = ADM.apeynom,
        ConsorcioNombre = CON.nombre,
        Periodo         = G.periodo,
        fechapago       = G.fechapago,
        TipoGasto       = TG.descripcion
    FROM [dbo].[gasto] G
    INNER JOIN [dbo].[tipogasto] TG
        ON G.idtipogasto = TG.idtipogasto
    INNER JOIN  [dbo].[consorcio] CON
        ON G.idconsorcio = CON.idconsorcio
        AND G.idprovincia = CON.idprovincia
        AND G.idlocalidad = CON.idlocalidad
    INNER JOIN [dbo].[administrador] ADM
        ON CON.idadmin = ADM.idadmin
)
GO
```

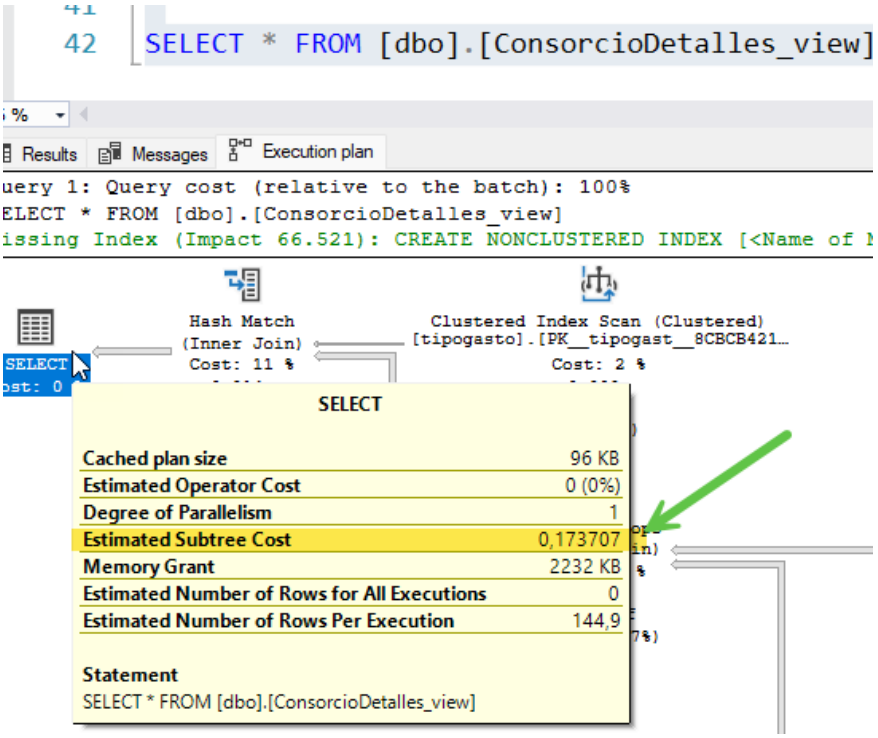
Modificar ConsorcioDetalles_view para pasar a vista indexada.

Procedemos a agregar un índice clúster único (Un índice clúster es un índice que determina el orden físico de los datos en una tabla o vista.)

```
-- Crear el indice
CREATE UNIQUE CLUSTERED INDEX [ix_idGasto] ON [dbo].[ConsorcioDetalles_view]
(
    [IdGasto]
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF)

```

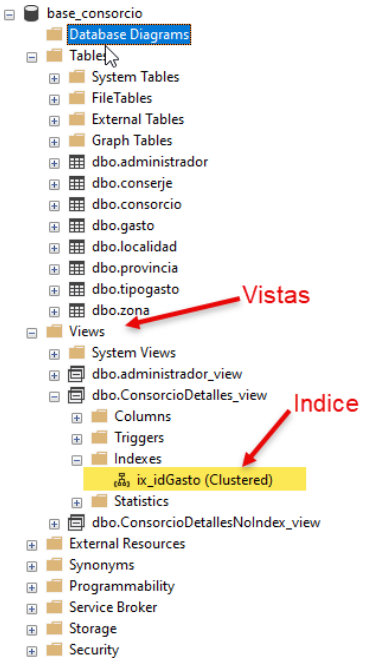
Aunque hemos obtenido el mismo resultado de la consulta anterior, hemos mejorado sustancialmente el rendimiento en la velocidad de búsqueda de los datos para la vista ahora indexada por el IdGasto. Pero ¿Cómo podemos observar dicha velocidad? Esto lo podemos observar si volvemos a ejecutar la consulta `SELECT` a la vista de `consorcioDetalles_view` , pero esta vez viendo el plan de ejecución que nos proporciona SQL Server.



Observación de plan de ejecución en la vista con índice cláusterizado

El valor del "Estimated Subtree Cost" en SQL Server no se mide en unidades específicas como KB, MB o GB. Es una unidad de medida relativa y adimensional que se utiliza para comparar la eficiencia relativa de diferentes operadores o subárboles en el plan de ejecución de una consulta.
Como se puede apreciar, el costo es relativamente bajo y esto contribuye a que la consulta ahora sea más eficiente en términos de recursos de CPU y E/S. Un valor más bajo indica un operador o subárbol más eficiente en términos de costo estimado.

Las vistas creadas se encuentran disponibles en SQL Server siguiendo el siguiente esquema mostrado a continuación:



Esquema de vistas y índice clausterizado para la DB base_consorcio

En la vista anterior, al momento de crear el índice clúster único, hemos agregado además unas cláusulas adicionales que nos proporciona SQL Server para poder gestionar de manera mas eficiente e integra el índice cláusterizado, las definiciones son variadas y se pueden implementar de acuerdo con los requerimientos necesarios para la DB, performance o seguridad de la vista.

Cláusulas para indexación de vistas

También podemos agregar cláusulas a un índice de una vista, la elección de los valores dependerá de los requisitos específicos y de la naturaleza de los datos y consultas que se ejecuten en la vista indexada, con el fin de considerar el rendimiento, el espacio en disco y otros factores al configurar estas opciones.

1. **PAD_INDEX = OFF**: Esta cláusula controla si se rellena la página de índice con valores de relleno para mantener un espacio constante en cada página. Cuando está configurado en "OFF", no se rellenan páginas de índice. Esto puede ahorrar espacio en disco, pero puede causar cierta fragmentación en el índice.
2. **STATISTICS_NORECOMPUTE = OFF**: Esta cláusula controla si se deben actualizar las estadísticas del índice automáticamente. Cuando está configurado en "OFF", SQL Server actualizará automáticamente las estadísticas del índice para mejorar el rendimiento de las consultas.

3. **SORT_IN_TEMPDB = OFF**: Esta cláusula controla si el proceso de creación del índice se realiza en la base de datos TEMPDB o en la base de datos actual. Cuando está configurado en "OFF", el proceso se realiza en la base de datos actual.
4. **IGNORE_DUP_KEY = OFF**: Esta cláusula controla si se deben permitir valores duplicados en el índice. Cuando está configurado en "OFF", no se permiten valores duplicados y se generará un error si se intenta insertar un valor duplicado.
5. **DROP_EXISTING = OFF**: Esta cláusula controla si se debe eliminar un índice existente con el mismo nombre antes de crear la vista indexada. Cuando está configurado en "OFF", no se eliminará un índice existente y se generará un error si ya existe un índice con el mismo nombre.

Requisitos adicionales

También se deben cumplir los siguientes requisitos, además de las opciones SET y los requisitos de función deterministas.

- El usuario que ejecuta `CREATE INDEX` debe ser el propietario de la vista.
- La vista se debe crear mediante la opción `WITH SCHEMABINDING`.
- La vista solo debe hacer referencia a tablas base que estén en la misma base de datos que la vista. La vista no puede hacer referencia a otras vistas.
- Si `GROUP BY` está presente, la definición de `VIEW` debe contener `COUNT_BIG(*)`, pero no `HAVING`. Estas restricciones `GROUP BY` solo se pueden aplicar a la definición de vista indizada. Una consulta puede usar una vista indexada en su plan de ejecución, aunque no cumpla estas restricciones `GROUP BY`.
- Al crear el índice, la opción de índice `IGNORE_DUP_KEY` debe establecerse en `OFF` (la configuración predeterminada).
- En la definición de vista, se debe hacer referencia a las tablas mediante nombres de dos partes, esquema.nombredetabla.
- Las funciones definidas por el usuario a las que se hace referencia en la vista se deben crear con la opción `WITH SCHEMABINDING`.
- Para hacer referencia a las funciones definidas por el usuario a las que se hace referencia en la vista, se deben usar nombres de dos partes, <schema>.<function>.

Función de Transact-SQL	**Posibles alternativas**
`COUNT`	Usar `COUNT_BIG`
Funciones `ROWSET` (`OPENDATASOURCE`, `OPENQUERY`, `OPENROWSET` y `OPENXML`)	
Media aritmética `AVG`	Usar `COUNT_BIG` y `SUM` como columnas independientes

Funciones de agregado estadístico (`STDEV`, `STDEVP`, `VAR` y `VARP`)	
Función `SUM` que hace referencia a una expresión que acepta valores `NULL`	Usar `ISNULL` dentro de `SUM()` para que la expresión no acepte valores `NULL`
Otras funciones de agregado (`MIN`, `MAX`, `CHECKSUM_AGG` y `STRING_AGG`)	
Funciones de agregado definidas por el usuario (SQL CLR)	

Tabla de alternativas para el uso de funciones en vistas indexadas.

Diferencia entre Vistas y Vistas indexadas:

Vistas	Vistas indexadas
● Una vista es una consulta almacenada que se guarda con un nombre y se puede utilizar como una tabla virtual. ● No almacena datos físicamente; en cambio, proporciona una forma de ver los datos de una o varias tablas, o incluso otras vistas. ● Las vistas no tienen almacenamiento asociado y siempre muestran datos en tiempo real según la consulta subyacente. ● Pueden contener joins, funciones, y cláusulas WHERE para filtrar los resultados.	● También conocidas como vistas indexadas materializadas, son vistas que tienen un índice cluster o non-cluster asociado, lo que permite almacenar físicamente los datos resultantes de la vista. ● Almacenan datos en disco, lo que puede mejorar el rendimiento de las consultas que utilizan la vista, especialmente en casos donde la consulta es compleja y costosa. ● Debido a que los datos se almacenan físicamente, las vistas indexadas pueden mejorar el rendimiento en situaciones donde las consultas son comunes pero los datos cambian con menos frecuencia. ● Las vistas indexadas tienen limitaciones en términos de qué tipo de consultas se pueden materializar y qué operaciones se pueden realizar en las tablas subyacentes.

Algunas diferencias clave a considerar son:

Rendimiento: Las vistas indexadas pueden mejorar el rendimiento en comparación con las vistas normales, ya que almacenan datos físicamente y pueden utilizar índices.

Actualización de Datos: Las vistas indexadas pueden tener limitaciones en cuanto a la capacidad de realizar actualizaciones en las tablas subyacentes. Pueden requerir esfuerzos adicionales para mantener la consistencia de los datos.

Uso de Recursos: Dado que las vistas indexadas almacenan datos físicamente, ocupan espacio en disco. Además, suelen requerir más recursos para mantener la consistencia con los datos subyacentes.

Cómo agregar permisos a una vista

Para empezar, recordemos que debemos tener primeramente una view creada, a partir de allí comenzaremos la realización de asignación de permisos de usuario. siguiendo el siguiente camino:

1. Crearemos 2 usuarios a nivel de servidor (por ende no ejecutamos la query en la db de base_consorcio) en cambio usaremos la base por defecto llamada master.

```
5   --Usuario administrador
6   CREATE LOGIN UsuarioAdmin WITH PASSWORD = 'admin123';
7   --Usuario de Solo Lectura
8   CREATE LOGIN UsuarioObservador WITH PASSWORD = 'observador123';
```

2. Luego, creamos los usuarios a nivel de base y asignamos el login y los permisos a los usuarios creados, uno como Admin y otro como Solo observador “Se entiende por observador que solo puede realizar consulta de tipo (SELECT)”

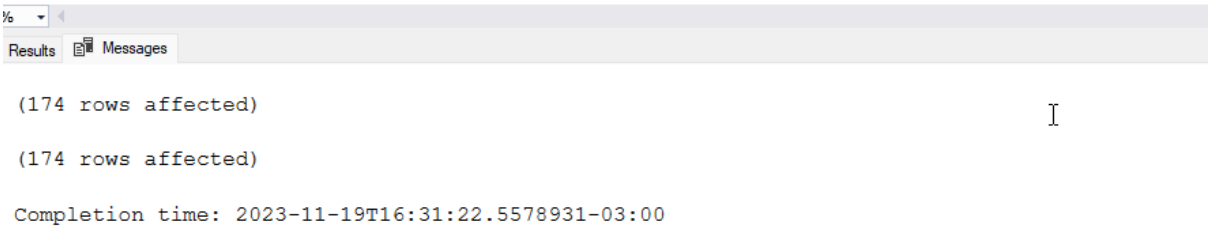
```
-- Se asigna el permiso de Admin para UsuarioAdmin
USE [base_consorcio];
CREATE USER UsuarioAdmin FOR LOGIN UsuarioAdmin;
ALTER ROLE db_owner ADD MEMBER UsuarioAdmin

-- Con la palabra 'GRANT' asignamos el permiso SELECT sobre la view [administrador_view]
CREATE USER UsuarioObservador FOR LOGIN UsuarioObservador;
GRANT SELECT ON [dbo].[administrador_view] TO UsuarioObservador;
```

3. Procedemos a probar con los usuarios nuevos la consulta SELECT, en ambos casos la solicitud debe ser correcta y arrojar resultados.

```
--Consulta SELECT con el usuario administrador
EXECUTE AS LOGIN = 'UsuarioAdmin'
SELECT * FROM [dbo].[administrador_view]
REVERT --REVERT en SQL Server se utiliza para volver al contexto original

--Consulta SELECT con el usuario observador
EXECUTE AS LOGIN = 'UsuarioObservador'
SELECT * FROM [dbo].[administrador_view]
REVERT
```



4. Ahora, probemos que el rol de observador no pueda realizar otras acciones fuera del SELECT, probemos por ejemplo el UPDATE.

```
--Al intentar realizar un UPDATE con el 'UsuarioObservador' es rechazado.
USE [base_consorcio];
EXECUTE AS LOGIN = 'UsuarioObservador';
UPDATE [dbo].[administrador_view]
   SET [sexo] = 'F'
 WHERE apeynom LIKE 'ESPINOZA JULIO'
GO
SELECT * FROM [dbo].[administrador_view]
REVERT;
```

```
Results  Messages
Msg 229, Level 14, State 5, Line 32
Se denegó el permiso UPDATE en el objeto 'administrador_view', base de datos 'base_consorcio', esquema 'dbo'.
```

Como puede apreciarse, la asignación de usuarios con permisos en las view, nos permite restringir ciertas funciones a los usuarios que estén limitados a realizar cualquier modificación y/o consulta que no se encuentre bajo su rol.

Cómo agregar Indices a las vistas

Existen diferentes tipos de índices de los cuales se ha realizado la implementación de 4 tipos en vistas de tipo indexado

Índices agrupados:

Un índice clúster ordena y almacena las filas de datos de la tabla o vista por orden en función de la clave del índice clúster. El índice clúster se implementa como una estructura de árbol b que admite la recuperación rápida de las filas a partir de los valores de las claves del índice clúster

Las características que poseen estos tipos de índices son:

- Los datos se ordenan físicamente por la clave que tiene el índice

- Solo se admite un índice por tabla lo que hace que sea crítico determinar cual sea para el rendimiento
- Dado que las filas de datos se almacenan en el orden del índice, las consultas que utilizan la clave del índice agrupado pueden ser más rápidas, ya que se reduce la necesidad de buscar en otros índices o la tabla en sí.

Ventajas

- Mejora el rendimiento de las consultas a través de los índices
- No se necesitan datos adicionales para localizar la fila una vez encontrada
- Se acelera la recuperación de valores de rangos de valores basados en las claves de índice agrupado

Desventajas

- Las operaciones CRUD son más lentas debido a que se debe reorganizar todo para mantener el orden de índice
- Si las operaciones son frecuentes se puede generar una fragmentación o corrupcion

```
-- Crear el indice, en este caso el indice es de tipo agrupado, ya que ordena los datos en base a la clave que toma el indice

CREATE UNIQUE CLUSTERED INDEX [ix_idGasto] ON [dbo].[ConsorcioDetalles_view]
(
    [IdGasto]
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF);
```

<u>Índice no agrupado:</u>

Los índices no agrupados o montones cada fila del índice no clúster contiene un valor de clave no agrupada y un localizador de fila. Este localizador apunta a la fila de datos del índice clúster o el montón que contiene el valor de clave. Las filas del índice se almacenan en el mismo orden que los valores de la clave del índice, pero no se garantiza que las filas de datos estén en un determinado orden a menos que se cree un índice clúster en la tabla

Características de los índices no agrupados

- No afecta al orden físico, ya que tienen apuntadores a las filas de datos
- Debido a que son estructuras que contienen apuntadores, las búsquedas son más rápidas
- Se pueden tener múltiples consultas de índices no agrupadas lo que permite indexar diferentes columnas para mejorar el rendimiento

Ventajas

- Se evita tener que recorrer la tabla completa en las consultas
- Las operación CRUD son más rápidas que las que tienen índices agrupados

Desventajas

- Una vez encontrado el índice, se debe buscar nuevamente en la tabla para recuperar los datos
- Ocupa más espacio el hecho de tener una estructura de índice separada

```
--Crear indice no agrupado, o monton; cada fila contiene un valor de clave no agrupada y un localizador de fila, este localizador
  apunta a la fila de datos del indice cluster o el monton que contiene el valor de clave

CREATE NONCLUSTERED INDEX [IX_NonClustered] ON [dbo].[ConsorcioDetalles_view] ([IdGasto]) WITH (PAD_INDEX = OFF,
  STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF);
```

Índice único:

Un índice único se asegura de que la clave de índice no contenga valores duplicados y, por tanto, cada fila de la tabla o vista sea en cierta forma única. La unicidad puede ser una propiedad tanto de índices clúster como de índices no clúster

Características de los índices únicos:

- Unicidad de los valores, no existen índices duplicados
- Se asemejan a los índices no agrupados, pero garantizan la unicidad
- Se pueden tener muchos índices únicos en una sola tabla

Ventajas de los índices únicos

- Evitar datos duplicados
- Mejora la integridad de los datos y facilita la aplicación de reglas de negocio con valores únicos

Desventajas

- Las operaciones de inserción se ralentiza ya que se debe verificar la unicidad de los datos
- La estructura de índice ocupa espacio adicional

```
--Crear indice Unico, se usa para que la clave de indice no contenga valores duplicados y para que cada fila o vista sea unica, y
  aplica tanto para indices cluster como los no cluster

CREATE UNIQUE INDEX [IX_Unique] ON [dbo].[ConsorcioDetalles_view] ([IdGasto]) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
  SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF);
```

Índice Filtrado:

Este tipo de índice se crea utilizando una cláusula WHERE en su definición para filtrar qué filas de la tabla deben incluirse en el índice. Un índice filtrado bien diseñado puede mejorar el rendimiento de las consultas y reducir los costos de almacenamiento del índice en relación con los índices de tabla completa, así como los costos de mantenimiento.

Características de los índices filtrados:

- No se usan todas las filas de la tabla al contener una cláusula where que permite la filtración de datos
- Al reducir el número de columnas se mejora significativamente el rendimiento de las consultas
- Los índices reducidos de los subconjuntos reducen el espacio ocupado

Ventajas

- Mejora el rendimiento al indexar sólo un grupo de filas
- Reduce el tamaño del índice y el uso de espacio en disco al almacenar menos datos

Desventajas

- Los índices filtrados pueden aumentar la complejidad del mantenimiento del esquema, que debe estar en un constante y correcto mantenimiento

```
--Crear indice filtrado, es uno que no se encuentra optimizado, se usa para realizar consultas en un conjunto reducido de datos y se
  usa junto con una clausula WHERE en su definicion para filtrar que filas de la tabla deben incluirse en el indice, dependiendo del
  diseño del indice tambien vendra el rendimiento de las consultas y la reduccion de almacenamiento

CREATE NONCLUSTERED INDEX [IX_Filtered] ON [dbo].[ConsorcioDetalle_view] ([IdGasto]) WHERE [IdGasto] > 15;
```

Otros tipos de índices:

Índice con columnas

Son una extensión de los índices no agrupados que permiten la inclusión de columnas adicionales no clave dentro de la estructura del índice. Estas columnas no clave incluidas son parte del índice, pero no se utilizan para ordenar o limitar la unicidad en el índice. En cambio, están disponibles para mejorar el rendimiento de consultas específicas. Las mejoras en el rendimiento se consiguen porque el optimizador de consultas puede localizar todos los valores de las columnas del índice, sin tener acceso a los datos de la tabla o del índice clúster, lo que da como resultado menos operaciones de E/S de disco

Características de los índices con columnas

- Columnas adicionales en el índice además de las columnas clave del índice, las columnas no clave pueden incluirse en la estructura del índice para mejorar el rendimiento de consultas.
- No se usan para ordenar, se usan para las eficiencias de consulta
- Mejora el rendimiento de consultas específicas en los que se necesitan ciertos tipos de datos

Ventajas

- Mejora el rendimiento de consultas al proporcionar cobertura y evitar búsquedas adicionales en la tabla de datos
- Optimiza el rendimiento sin aumentar la cantidad de índices ya que las columnas forman parte del índice agrupado

Desventajas

- Aumenta el tamaño del índice y con eso la cantidad de espacio usado en el disco
- Son de uso estratégico y específico

CAPÍTULO V: CONCLUSIONES

Las vistas en SQL Server son útiles para simplificar y abstraer la complejidad de las consultas, permitiendo una mejor organización del código SQL y el acceso a datos. Además, las vistas indexadas pueden mejorar significativamente el rendimiento de consultas complejas, especialmente cuando se utilizan índices adecuados. Sin embargo, es importante usarlas con moderación y entender su impacto en el rendimiento, ya que la elección de índices y la optimización de consultas son críticas. Las vistas indexadas son una herramienta poderosa cuando se utilizan de manera adecuada.

CAPITULO VI: BIBLIOGRAFÍA

Microsoft Ignite, 18/08/2023, Creación de Vistas, manejo de vistas aplicadas en Sql Server, Sql Azure server y Azure Sql Managed Instance.
SqlLearning, 2023, VISTAS EN SQL, (Create, Alter and DROP) en vistas de SQL.
Newsmactic, 2021, Qué es una vista indexada y cómo mejorar el rendimiento de consultas en SQL Server, definiciones e implementación de vistas indexadas.
il_masacratore, 11/02/2013, SQL Server: Vistas indizadas y el porqué de usarlas para cargas de dwh, Implementación de vistas indizadas en Sql Server.