Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura
Licenciatura en Sistemas de Información
Base de datos I
Año: 2023

Índices columnares SQL

Comisión 1 grupo 5

Alumno: Bournisent, Matias
L.U. N°: 50.434
D.N.I. N°: 39.126.075

Alumno: Comba, Carlos Alfredo
L.U. N°: 44.055
D.N.I. N°: 32.335.478

Alumno: Ramirez Gonzalo Daniel
L.U. N°: 56.838
D.N.I. N°: 44.543.439

Alumno: Zini, Franco Joaquin
L.U. N°: 50.717
D.N.I. N°: 40.049.028

Índice

Capítulo I: Introducción

En el ámbito de la materia Base de Datos I, se nos asignó la tarea de investigar y experimentar con un componente fundamental del diseño de bases de datos: **los índices columnares**. El enfoque se centra en analizar cómo los índices impactan en el rendimiento del sistema, utilizando un caso de estudio concreto.

Partiendo de un modelo de base de datos proporcionado por la cátedra, llamado "base_consorcio", nuestra tarea es duplicar la tabla 'gasto', creando así una nueva tabla denominada 'gastoNew'. La clave de nuestro estudio es la implementación de índices columnares en esta nueva tabla, lo que nos permitió investigar cómo estos índices aceleran la recuperación de datos.

Ambas tablas se cargaron con un millón de registros para crear un escenario realista de carga de trabajo. Nuestro objetivo es evaluar y comparar las diferencias de rendimiento entre las búsquedas realizadas en una tabla con índices columnares ('gastoNew') y una sin ellos ('gasto'), para entender la mejora que ofrecen estos índices.

Este informe proporcionará detalles de nuestra metodología, incluyendo los fundamentos teóricos de los índices columnares, los scripts utilizados para modificar las tablas y realizar las inserciones, así como los scripts de búsqueda. Además, se detalla el entorno técnico en el que se llevaron a cabo las pruebas, incluyendo el motor y gestor de la base de datos utilizados, así como las especificaciones de las máquinas que alojan nuestras pruebas.

Los resultados de nuestras pruebas consisten en los tiempos de respuestas de cada consulta, así como gráficas comparativas que permiten ver la mejora de tiempo utilizando los índices columnares. Estos datos permiten cuantificar el impacto de los índices columnares en el rendimiento del sistema.

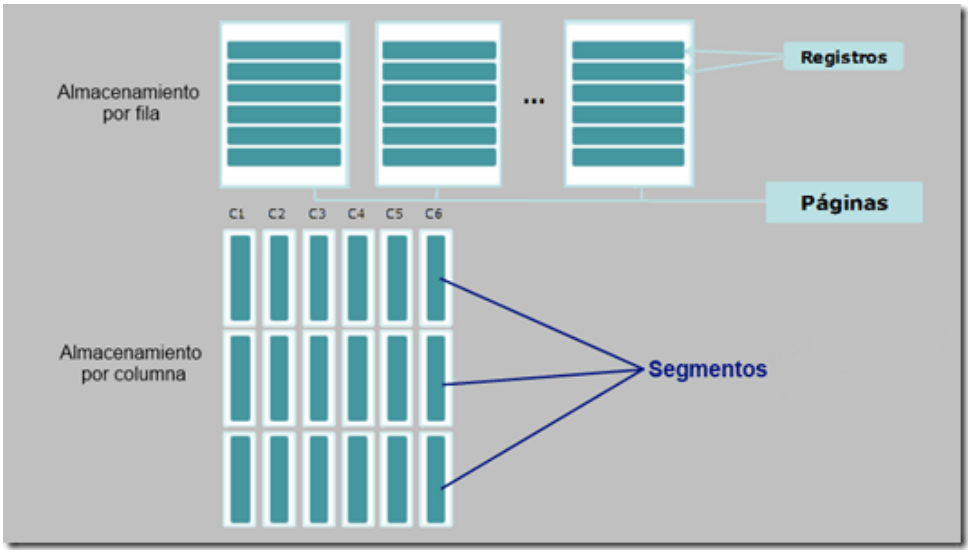
Capítulo II: Marco teórico

En la gestión de bases de datos, el uso de índices columnares optimiza el rendimiento. Son estructuras de datos especializadas que permiten una recuperación eficiente de información en grandes conjuntos de datos. En lugar de buscar a través de todas las filas de una tabla, los índices columnares almacenan y ordenan los valores de una columna específica, actuando como rutas de acceso rápidas hacia los datos que los usuarios buscan. Markus Winand nos dice que "Los índices columnares son un tipo de índice que almacena los datos en columnas, en lugar de filas. Este enfoque de almacenamiento permite que el motor de base de datos acceda a los datos de manera más eficiente, lo que puede traducirse en un rendimiento de las consultas más rápido." (SQL Performance Explained" por Markus Winand, pag. 102)

Siguiendo con los conceptos de Markus Winand nos explica también que "Los índices columnares son particularmente útiles para consultas que requieren operaciones de agrupación, ordenación o filtros complejos. En estas situaciones, los índices columnares pueden proporcionar un rendimiento de las consultas significativamente mejor que los índices tradicionales." (SQL Performance Explained" por Markus Winand, pag. 102)

En el mismo libro encontramos un concepto más complejo sobre las teorías de "Los índices columnares que utilizan técnicas de compresión para acelerar las consultas. Estas técnicas permiten al motor de base de datos acceder a los datos de manera más eficiente."... "Las técnicas de compresión de índices columnares se pueden utilizar para reducir el tamaño de los datos almacenados en el índice. Esto puede mejorar el rendimiento de las consultas, ya que el motor de base de datos no tiene que leer tantos datos para encontrar los registros que coinciden con la consulta." ((Fundamentos de Base de Datos" de Korth H. y Silberschatz A. pág. 381-382)

El almacenamiento columnar se realiza de la siguiente forma:



- En primer lugar, consideremos el enfoque tradicional de almacenamiento por filas, donde los datos se organizan en páginas de datos y se almacenan fila por fila en estas páginas.
- En contraste, en el almacenamiento por columnas, se adopta un enfoque diferente. Aquí, los valores de cada columna se separan y almacenan en segmentos individuales. Estos segmentos contienen los datos de las columnas y se someten a compresión utilizando la tecnología VertiPaq(Tecnología de compresión de datos).

Cuando se crea un índice columnar para una columna específica, la base de datos organiza los valores de esa columna en una estructura de datos optimizada:

- **Árboles B:** Los índices columnares basados en árboles B son estructuras de datos de tipo árbol que permiten búsquedas eficientes, inserciones y eliminaciones. Estos árboles están diseñados para minimizar la altura del árbol, lo que vuelve la búsqueda más rápida.

- **Árboles B+:** Son una variante de los árboles B que mejora aún más la eficiencia en las operaciones de búsqueda. Los nodos internos de un árbol B+ contienen solo claves para la navegación, lo que aumenta la capacidad de almacenamiento de las claves y mejora la velocidad de búsqueda.

- **Estructuras de Bitmap:** Los índices columnares basados en estructuras de bitmap son eficaces para columnas que tienen un número limitado de valores únicos (por ejemplo, género o estado civil). Cada valor único se asigna a un conjunto de bits, y las operaciones de búsqueda implican operaciones lógicas en estos conjuntos de bits, lo que permite búsquedas rápidas y eficientes.

- **Índices Hash:** Los índices hash se basan en funciones hash que asignan valores de columna a ubicaciones específicas en una estructura de datos hash. Esto permite búsquedas rápidas cuando se conoce el valor exacto de la columna. Sin embargo, los índices hash no son tan eficientes para operaciones como el rango o las comparaciones de patrones.

- **Árboles Trie:** Estos árboles son útiles para índices columnares en datos de texto o cadenas. Los árboles Trie almacenan información de texto de manera jerárquica, lo que facilita la búsqueda y comparación de palabras o patrones de texto.

La elección de la estructura asociada a un índice columnar depende del tipo de datos en la columna y del tipo de operaciones que se realizarán con esa columna (por ejemplo, búsquedas exactas, búsquedas de rango, etc.). Cada estructura tiene sus ventajas y desventajas, por lo que se selecciona según los requisitos específicos de la aplicación. El motor de base de datos toma la decisión de qué

estructura de índice utilizar. Los DBMS están diseñados para ser sistemas inteligentes y autónomos que optimizan el rendimiento de las consultas y operaciones en función de la estructura de los datos y el tipo de consultas que se realizan.
Ventajas de los índices columnares:
Los índices columnares ofrecen una serie de ventajas sobre los índices tradicionales, entre las que se incluyen:

- Mejor rendimiento para consultas que requieren la comparación de valores en una sola columna.
- Reducción del uso de memoria del servidor.
- Mejor compresión de datos.
- Reducción de la fragmentación de datos.

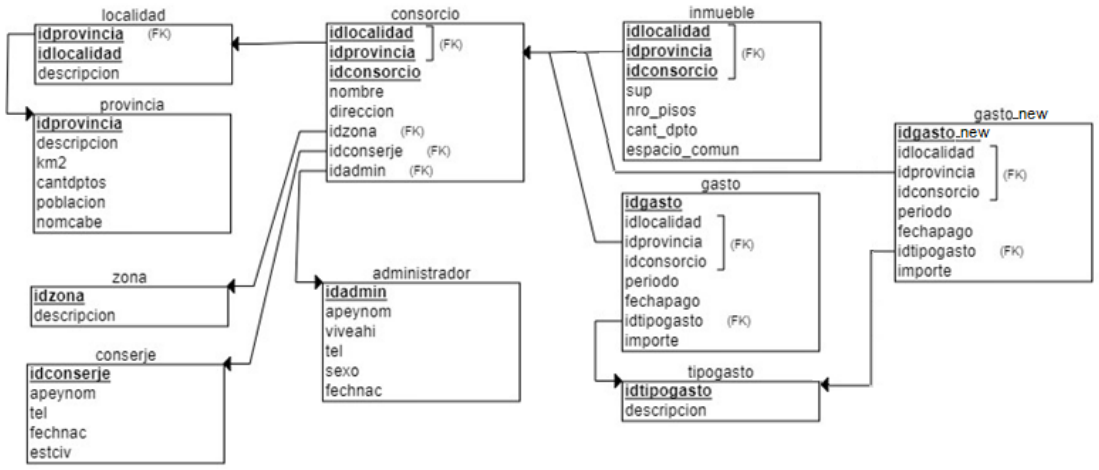
Los índices columnares no son ideales para todos los entornos, por ejemplo estos no son ideales en el ámbito de procesamiento de transacciones en línea (OLTP) donde se realizan operaciones de inserción, actualización y eliminación con alta frecuencia. Están orientados para las situaciones en las que las operaciones de lectura son más frecuentes comparadas a las de escritura. Otra desventaja que presentan Los índices columnares es que pueden requerir más espacio de almacenamiento que los índices tradicionales basados en filas, especialmente durante el proceso de carga de datos, debido a esto el costo de almacenamiento puede incrementarse enormemente, también el uso de índices columnares puede requerir recursos adicionales del sistema, como CPU y memoria, para su procesamiento. En entornos con recursos limitados, esto puede ser un factor a considerar.

Los índices columnares se utilizan a menudo en situaciones donde se requiere un alto rendimiento para consultas analíticas y agregaciones en bases de datos que manejan grandes volúmenes de datos, las más populares son:

- Análisis de datos (OLAP)
- Data warehousing:
- Business intelligence
- Big data

Capítulo III: Metodología

Modelo sobre que se va a trabajar



diccionario de datos

Localidad

CAMPO	TIPO	DESCRIPCIÓN
idprovincia	int	Referencia a la provincia a la que pertenece la localidad
idlocalidad	int	Identificador único dentro de la provincia
descripcion	varchar(50)	Nombre de la localidad. Puede ser nulo.

Provincia

CAMPO	TIPO	DESCRIPCIÓN
idprovincia	int	Identificador único de la provincia
descripcion	varchar(50)	Nombre de la provincia. Puede ser nulo.

km2	int	Superficie de la provincia en kilómetros cuadrados. Puede ser nulo.
cantdptos	int	Cantidad de localidades que posee la provincia. Puede ser nulo.
poblacion	int	Cantidad de habitantes que posee la provincia. Puede ser nulo.
nomcabe	varchar(50)	Nombre de la capital de la provincia. Puede ser nulo.

Zona

CAMPO	TIPO	DESCRIPCIÓN
idzona	int	Identificador único de la zona
descripcion	varchar(50)	Nombre de la zona. Puede ser nulo.

Conserje

CAMPO	TIPO	DESCRIPCIÓN
idconserje	int	Identificador único del conserje.
apeynom	varchar(50)	Apellido y nombre del conserje. Puede ser nulo.
tel	varchar(20)	Teléfono del conserje. Puede ser nulo.
fechnac	datetime	Fecha de nacimiento del conserje. Puede ser nulo.

estciv	varchar(1)	Estado civil del conserje. Puede ser nulo. Valores posibles: - S - Soltero - C - Casado - D - Divorciado - O - Otro

Consorcio

CAMPO	TIPO	DESCRIPCIÓN
idprovincia	int	Referencia a la provincia a la que pertenece el consorcio
idlocalidad	int	Referencia, junto con el idprovincia, a la localidad a la que pertenece el consorcio
idconsorcio	int	Identificador único del consorcio dentro de la localidad
nombre	varchar(50)	Nombre del consorcio. Puede ser nulo.
direccion	varchar(250)	Dirección en la que se encuentra ubicado el consorcio. Puede ser nulo.
idzona	int	Referencia a la zona en donde está ubicado el consorcio. Puede ser nulo.
idconserje	int	Referencia al conserje asignado al consorcio. Puede ser nulo.
idadmin	int	Referencia al administrador asignado al consorcio. Puede ser nulo.

Administrador

CAMPO	TIPO	DESCRIPCIÓN
idadmin	int	Identificador único del administrador
apeynom	varchar(50)	Apellido y nombre del administrador. Puede ser nulo.
viveahi	varchar(1)	Indica si el administrador reside en el consorcio al que esta asignado. Puede ser nulo. Valores posibles: - N - No - S - Si
tel	varchar(20)	Teléfono del administrador. Puede ser nulo.
sexo	varchar(1)	Sexo del administrador. Valores posibles: - M - Masculino - F - Femenino
fechnac	datetime	Fecha de Nacimiento del administrador. Puede ser nulo.

Inmueble

CAMPO	TIPO	DESCRIPCIÓN
idprovincia	int	Referencia a la provincia a la que pertenece el inmueble
idlocalidad	int	Referencia, junto a idprovincia, a la localidad a la que pertenece el inmueble

idconsorcio	int	Referencia, junto a idprovincia e idlocalidad, al consorcio al que pertenece el inmueble
sup	decimal(6,2)	Superficie del inmueble en metros cuadrados. Puede ser nulo.
nro_pisos	int	Cantidad de pisos que posee el inmueble. Puede ser nulo.
cant_dpto	int	Cantidad de departamentos que posee el inmueble. Puede ser nulo.
espacio_comun	int	Cantidad de espacios comunes

Gasto

CAMPO	TIPO	DESCRIPCIÓN
idgasto	int	Identificador único del gasto
idprovincia	int	Referencia a la provincia donde esta situado el consorcio al que pertenece el gasto
idlocalidad	int	Referencia a la localidad, junto con idprovincia, donde está situado el consorcio al que pertenece el gasto
idconsorcio	int	Referencia al consorcio, junto con idprovincia e idlocalidad, al que pertenece el gasto
periodo	int	Número de mes del año al que pertenece el gasto. Puede ser nulo.
fechapago	datetime	Fecha en la que se realizó el pago del gasto. Puede ser nulo.

idtipogasto	int	Referencia al tipo de gasto. Puede ser nulo.
importe	decimal(8,2)	Importe del gasto. Puede ser nulo.

GastoNew

CAMPO	TIPO	DESCRIPCIÓN
idgasto_new	int	Identificador único del gasto
idprovincia	int	Referencia a la provincia donde esta situado el consorcio al que pertenece el gasto
idlocalidad	int	Referencia a la localidad, junto con idprovincia, donde está situado el consorcio al que pertenece el gasto
idconsorcio	int	Referencia al consorcio, junto con idprovincia e idlocalidad, al que pertenece el gasto
periodo	int	Número de mes del año al que pertenece el gasto. Puede ser nulo.
fechapago	datetime	Fecha en la que se realizó el pago del gasto. Puede ser nulo.
idtipogasto	int	Referencia al tipo de gasto. Puede ser nulo.
importe	decimal(8,2)	Importe del gasto. Puede ser nulo.

TipoGasto

CAMPO	TIPO	DESCRIPCIÓN
idtipogasto	int	Identificador único del tipo de gasto
descripcion	varchar(50)	Descripción del gasto.
		Puede ser nulo.

Proceso de trabajo

Sobre la base del modelo que entregó la cátedra. La estructura de base de datos y dos lotes de carga.
Donde se ejecuta:

1. ModeloDatos_Consorcio.sql: este script crea la base de datos y sus tablas.
2. BDI_loteDatosConsorcios.sql: permite cargar los datos de provincias, localidad, zona, conserje, administracion, consorcio, tipogasto, gasto.
3. BDI_tabla_inmueble_completo.sql: crea y carga la tabla inmueble.
4. columnstore_index_script.sq: crea la tabla gastonew, crea los índice columnares, se insertan 1 millón de registros, y se carga la tabla gasto con los registros faltantes para llegar al millón y poder hacer las pruebas comparativas.
5. script_busqueda.sql: Contiene el conjunto de pruebas.

Organización del grupo

Mediante reuniones sincrónicas realizamos la manera en la cual abordamos el proyecto sobre índices columnares, como así también creamos el repositorio en github para que cada alumno interactúe con el contenido, aportando sus resultados, scripts y para reunir toda la información recopilada por cada miembro.
Cada estudiante ejecutará el script de busqueda.sql y anotará sus resultados en la tabla de referencia. El test ejecuta 5 pruebas de distintas operaciones sobre ambas tablas, registrando el tiempo de ejecución en cada una

Herramientas

Elementos comunes a todos:

- Script en Transact-SQL (Es el lenguaje utilizado para interactuar y gestionar bases de datos en el sistema de gestión de bases de datos Microsoft SQL Server.)

Elementos usados por Comba Carlos para las pruebas:

- Motor de base de datos SQL Server 2022 standard versión 16.0.1000.6
- SQL management Studio 19.1.56
- Máquina i7 11gen, 32 gb ram, 1tb ssd.

Elementos de prueba usado por Ramirez Gonzalo para las pruebas:

- Motor de base de datos SQL Server 2022
- SQL management Studio 19.1.56
- Equipo: Ryzen 3 3100, 16 gb de ram, Ssd 240GB.

Elementos de prueba usado Zini Franco para las pruebas:

- Motor de base de datos SQL Server 2022
- SQL management Studio 19.1.56
- Equipo: i5 12gen, 16 gb de ram, ssd 480gb.

.Elementos de prueba usado por Bournisent Matias para las pruebas:

- Motor de base de datos SQL Server 2022
- SQL management t Studio 19.1.56
- Equipo: Ryzen 5 5500u, 8gb de ram,ssd 256gb

Procedimientos de Recopilación de Datos

Para obtener la información que necesitábamos para nuestro proyecto, principalmente lo obtuvimos de fuentes de paginas webs (aclaradas en sección de bliblografias). En este proceso, hicimos lectura de documentos técnicos, exploramos tutoriales y recursos que encontramos en línea. Este enfoque nos ayudó a reunir una variedad de datos útiles y nos dio una buena base para llevar a cabo nuestro proyecto.
La información y conocimiento adquirido por cada alumno fue puesta en escena en las reuniones realizadas, para así poder nutrirnos todos sobre el tema en cuestión y la realización del proyecto.

Capítulo IV: Desarrollo/Resultados

Introducción

En esta sección, haremos foco en el análisis de las pruebas de rendimiento realizadas en consultas que involucran el uso de índices columnares en comparación con aquellas que no los utilizan. Nuestra principal misión es evaluar y presentar resultados concretos que permitan comprender el impacto de estos índices en términos de tiempo, costos y rendimiento. Las pruebas son realizadas en distintos escenarios para así comprobar el desempeño de los índices columnares en situaciones variadas. Estos escenarios pueden incluir consultas que involucran un gran volumen de datos, consultas de agregación, búsquedas de información y más. Esto nos permitirá obtener una visión completa de cómo los índices columnares se desempeñan en diversas circunstancias.

Desarrollo

1. El primer paso crucial en nuestro proyecto fue la creación de la nueva tabla "gastoNew". Para garantizar la coherencia y la continuidad de los datos con respecto a la tabla existente "gasto", se optó por replicar cuidadosamente su estructura. Para ello, identificamos cada columna en la tabla "gasto" junto con su tipo de dato correspondiente, lo que nos permitió diseñar la estructura de "gastoNew" de manera precisa y efectiva.

```
CREATE TABLE [dbo].[gastonew](
    [idgasto] [int] IDENTITY(1,1) NOT NULL,
    [idprovincia] [int] NULL,
    [idlocalidad] [int] NULL,
    [idconsorcio] [int] NULL,
    [periodo] [int] NULL,
    [fechapago] [datetime] NULL,
    [idtipogasto] [int] NULL,
    [importe] [decimal](8, 2) NULL,
    CONSTRAINT [PK_gastonew] PRIMARY KEY CLUSTERED ([idgasto] ASC),
    CONSTRAINT [FK_gasto_consorcio_new] FOREIGN KEY ([idprovincia], [idlocalidad], [idconsorcio])
    REFERENCES [dbo].[consorcio] ([idprovincia], [idlocalidad], [idconsorcio]),
    CONSTRAINT [FK_gasto_tipo_new] FOREIGN KEY ([idtipogasto])
    REFERENCES [dbo].[tipogasto] ([idtipogasto])
);
```

2. Como segundo paso en nuestro proyecto, después de haber creado la tabla "gastoNew" basada en la estructura de la tabla "gasto", nos enfocamos en la creación del índice columnar. Diseñamos y creamos un índice columnar específico denominado "IX_GASTONEW_Columnar" en la tabla "gastoNew". Este índice se configuró para incluir columnas clave que son relevantes para las consultas financieras, como "idprovincia", "idlocalidad", "idconsorcio" etc. Cada una de estas columnas se almacenó y se indexó de manera

independiente lo que permitió una recuperación de datos más rápida y eficiente en comparación con los índices de filas tradicionales.

```
CREATE NONCLUSTERED COLUMNSTORE INDEX IX_GASTONEW_Columnar
ON dbo.GASTONEW
(
    idprovincia,
    idlocalidad,
    idconsorcio,
    periodo,
    fechapago,
    idtipogasto,
    importe
);
```

3. Después de haber creado la tabla "gastoNew" y el índice columnar, el tercer paso de nuestro proyecto involucró la generación de datos de prueba para llenar la tabla con información ficticia. Estos datos de prueba se utilizarían en nuestras pruebas de rendimiento y evaluación del índice columnar. Para facilitar la inserción de datos de prueba en la tabla, desactivamos temporalmente todas las restricciones de clave externa utilizando la sentencia SQL, Esta acción nos permitió insertar datos en la tabla sin preocuparnos por las restricciones de integridad referencial que normalmente se aplicarían.

```
ALTER TABLE dbo.gastonew NOCHECK CONSTRAINT ALL;

DECLARE @counter INT = 1;

WHILE @counter <= 1000000
BEGIN
    INSERT INTO dbo.gastonew (idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (
        -- Valores aleatorios para cada columna
        CAST((RAND() * (5-1) + 1) AS INT), -- idprovincia (aleatorio entre 1 y 5)
        CAST((RAND() * (5-1) + 1) AS INT), -- idlocalidad (aleatorio entre 1 y 5)
        CAST((RAND() * (40-1) + 1) AS INT), -- idconsorcio (aleatorio entre 1 y 400)
        CAST((RAND() * (9-1) + 1) AS INT), -- periodo (aleatorio entre 1 y 9)
        GETDATE(), -- fechapago (fecha y hora actual)
        CAST((RAND() * (5-1) + 1) AS INT), -- idtipogasto (aleatorio entre 1 y 5)
        CAST((RAND() * (400000-55000) + 55000) AS DECIMAL(8,2)) -- importe (aleatorio entre 50000 y 300000)
    );

    SET @counter = @counter + 1;
END;

-- Vuelve a habilitar las restricciones de clave externa
ALTER TABLE dbo.gastonew CHECK CONSTRAINT ALL;
GO
```

4. Como parte de nuestro enfoque en la comparación justa y precisa del impacto del índice columnar en las consultas, repetimos el proceso de generación de datos en la tabla "gasto". Esto nos permitió establecer una base de datos de pruebas similar en términos de volumen de datos y estructura de la tabla, lo que facilita la evaluación del rendimiento entre la tabla con índice columnar.

```
ALTER TABLE dbo.gasto NOCHECK CONSTRAINT ALL;

DECLARE @counter INT = 1;

WHILE @counter <= 992000
BEGIN
    INSERT INTO dbo.gasto (idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (
        -- Valores aleatorios para cada columna
        CAST((RAND() * (5-1) + 1) AS INT), -- idprovincia (aleatorio entre 1 y 5)
        CAST((RAND() * (5-1) + 1) AS INT), -- idlocalidad (aleatorio entre 1 y 5)
        CAST((RAND() * (40-1) + 1) AS INT), -- idconsorcio (aleatorio entre 1 y 400)
        CAST((RAND() * (9-1) + 1) AS INT), -- periodo (aleatorio entre 1 y 9)
        GETDATE(), -- fechapago (fecha y hora actual)
        CAST((RAND() * (5-1) + 1) AS INT), -- idtipogasto (aleatorio entre 1 y 5)
        CAST((RAND() * (400000-55000) + 55000) AS DECIMAL(8,2)) -- importe (aleatorio entre 50000 y 300000)
    );

    SET @counter = @counter + 1;
END;

-- Vuelve a habilitar las restricciones de clave externa
ALTER TABLE dbo.gasto CHECK CONSTRAINT ALL;
GO
```

5. Una vez que hemos completado los pasos anteriores, procedemos a ejecutar pruebas de consulta en las tablas "gastoNew" y "gasto". El objetivo de estas pruebas es comparar el tiempo de ejecucion de las consultas en ambas tablas para evaluar el impacto que tiene el índice columnar en el rendimiento de las operaciones.

Resultados de las pruebas

Después de ejecutar un conjunto de pruebas de consulta en las tablas "gastoNew" y "gasto", (Cuyas consultas se encuentran en el archivo busqueda.sql) hemos recopilado resultados significativos que nos permiten evaluar el impacto del índice columnar en el rendimiento de nuestras operaciones.
A Continuación cada alumno presenta la tabla de resultados de un total de 5 test, en donde se mide el tiempo de respuesta en milisegundos

Se ejecutaron los siguientes tests, con código adicional que mide el tiempo transcurrido, omitido aquí por brevedad, el código completo se encuentra en el repositorio en el archivo busqueda.sql:

```sql
--Test 1: Suma de la columna importe
SELECT @ValorSinColumnstore = SUM(importe) FROM gasto;
SELECT @ValorConColumnstore = SUM(importe) FROM gastonew;

--Test 2: promedio de la columna importe
SELECT @ValorSinColumnstore = AVG(importe) FROM gasto;
SELECT @ValorConColumnstore = AVG(importe) FROM gastonew;

--Test 3: Conteo de registros en cada tabla
SELECT @ValorSinColumnstore = COUNT(*) FROM gasto WHERE idconsorcio <= 250;
SELECT @ValorConColumnstore = COUNT(*) FROM gastonew WHERE idconsorcio <= 250;

--Test 4: Select * en cada tabla
SELECT * FROM gasto WHERE idconsorcio <= 250;
SELECT * FROM gastonew WHERE idconsorcio <= 250;

--Test 5: Select con join
SELECT gn.* , pro.* FROM gasto as gn
join provincia as pro on gn.idprovincia = pro.idprovincia
WHERE idconsorcio <= 250 and pro.idprovincia = 1

SELECT gn.* , pro.* FROM gastonew as gn
join provincia as pro on gn.idprovincia = pro.idprovincia
WHERE idconsorcio <= 250 and pro.idprovincia = 1
```

Al ejecutarse busqueda.sql se obtiene una salida similar a la siguiente:

Results | Messages

	Tabla	Cantidad	TiempoTranscurrido
1	TEST 1) SUM(importe) sin columnstore	227468207036	220

	Tabla	Cantidad	TiempoTranscurrido
1	TEST 1) SUM(importe) con columnstore	227309471498	180

	Tabla	Cantidad	TiempoTranscurrido
1	TEST 2) AVG(importe) sin columnstore	227468	220

	Tabla	Cantidad	TiempoTranscurrido
1	TEST 2) AVG(importe) con columnstore	227309	137

	Tabla	Cantidad	TiempoTranscurrido
1	TEST 3) COUNT(*) sin columnstore	1000000	93

	Tabla	Cantidad	TiempoTranscurrido
1	TEST 3) COUNT(*) con columnstore	1000000	107

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe
1	14528714	1	4	1	5	2023-10-30 15:32:19.170	4	236302.27

	Tabla	TiempoTranscurrido
1	TEST 4) SELECT sin columnstore	6790

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe
1	4992211	1	1	21	4	2023-10-29 01:26:05.637	3	333285.39

	Tabla	TiempoTranscurrido
1	TEST 4) SELECT con columnstore	6876

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe	idprovincia	descripcion	km2	cantdptos	poblacion	nomcabe
7	14528741	1	2	1	7	2023-10-30 15:32:19.177	1	80681.09	1	Capital Federal	203	1	2891082	Capital Federal

	Tabla	TiempoTranscurrido
1	TEST 5) SELECT/JOIN sin columnstore	2434

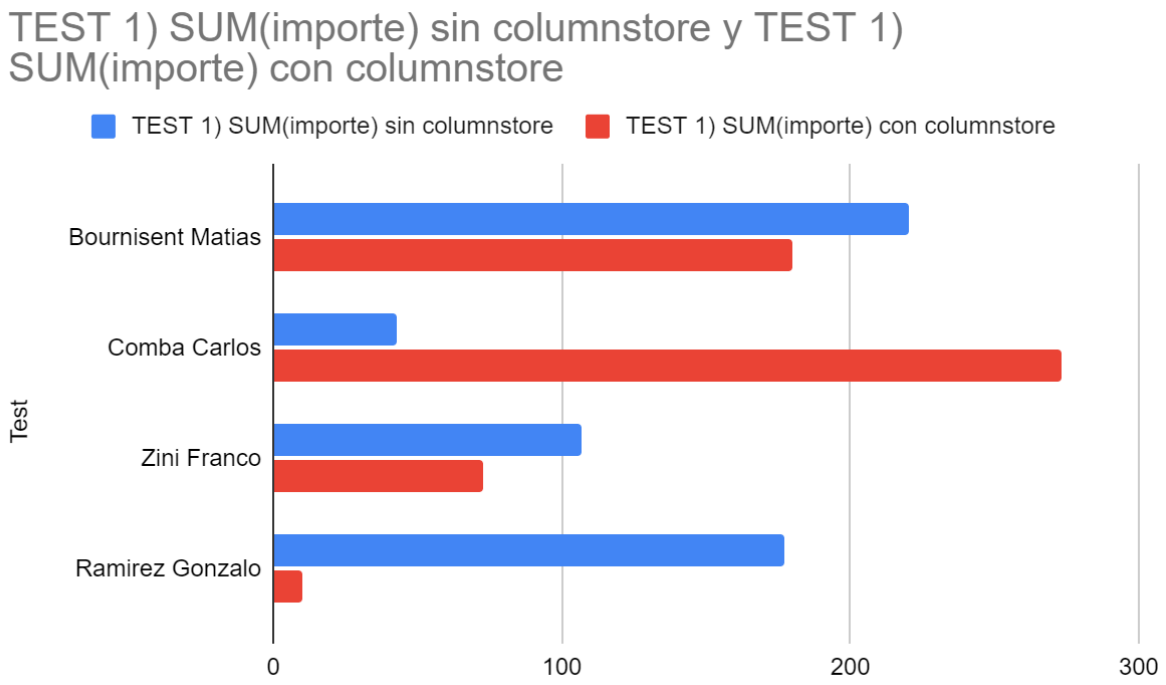
	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe	idprovincia	descripcion	km2	cantdptos	poblacion	nomcabe
1	4992211	1	1	21	4	2023-10-29 01:26:05.637	3	333285.39	1	Capital Federal	203	1	2891082	Capital Federal

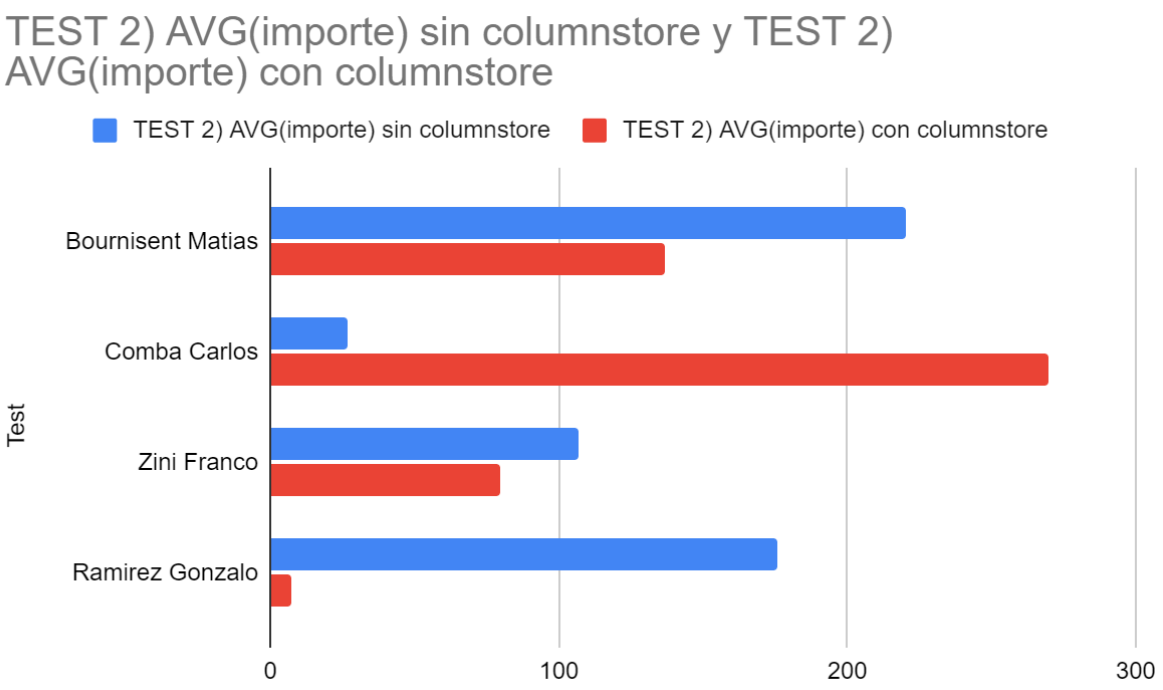
	Tabla	TiempoTranscurrido
1	TEST 5) SELECT/JOIN con columnstore	2516

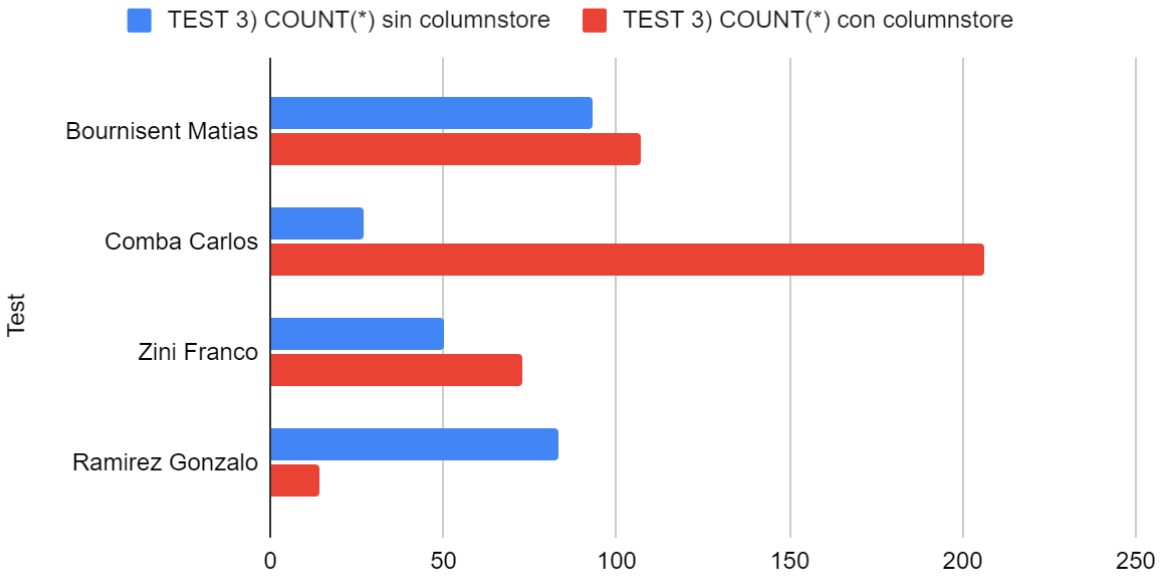
Los resultados de todos los integrantes fueron

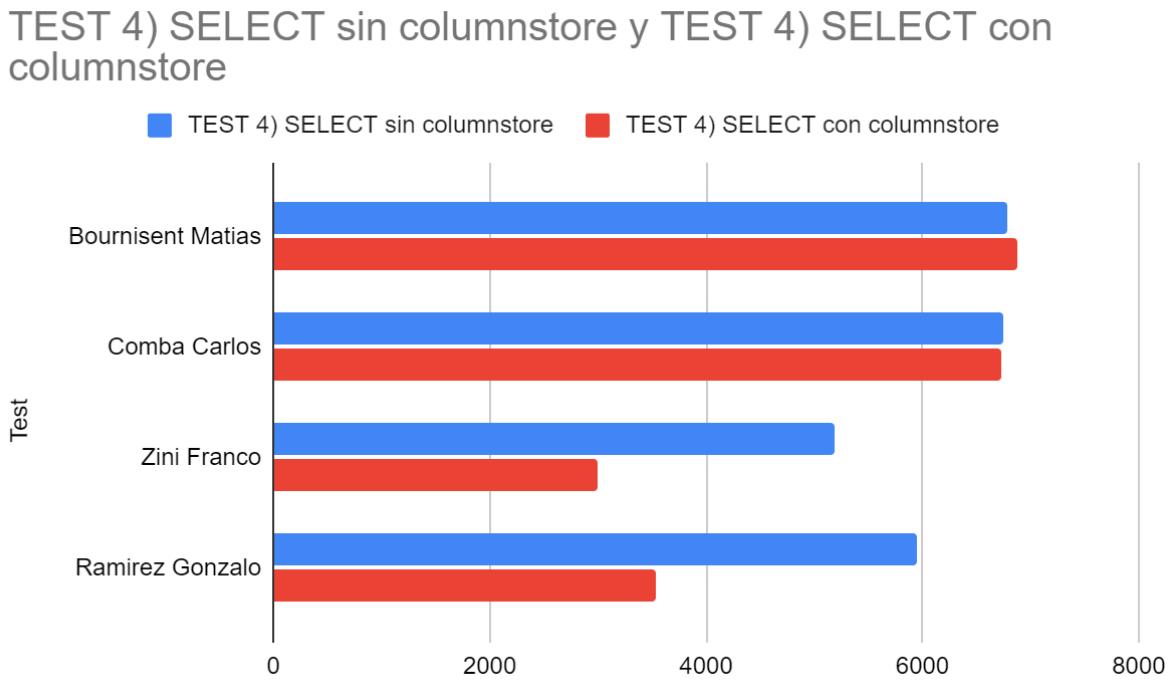
Test	Bournisent Matias	Comba Carlos	Zini Franco	Ramirez Gonzalo
TEST 1) SUM(importe) sin columnstore	*220*	*43*	*107*	*177*
TEST 1) SUM(importe) con columnstore	*180*	*273*	*73*	*10*
TEST 2) AVG(importe) sin columnstore	*220*	*27*	*107*	*176*
TEST 2) AVG(importe) con columnstore	*137*	*270*	*80*	*7*
TEST 3) COUNT(*) sin columnstore	*93*	*27*	*50*	*83*
TEST 3) COUNT(*) con columnstore	*107*	*206*	*73*	*14*
TEST 4) SELECT sin columnstore	*6790*	*6750*	*5197*	*5956*
TEST 4) SELECT con columnstore	*6876*	*6730*	*2990*	*3537*
TEST 5) SELECT/JOIN sin columnstore	*2434*	*2254*	*1096*	*1890*
TEST 5) SELECT/JOIN con columnstore	*2616*	*2286*	*1014*	*1186*
Cantidad de registros en las pruebas	1000000	1000000	1000795	1000000

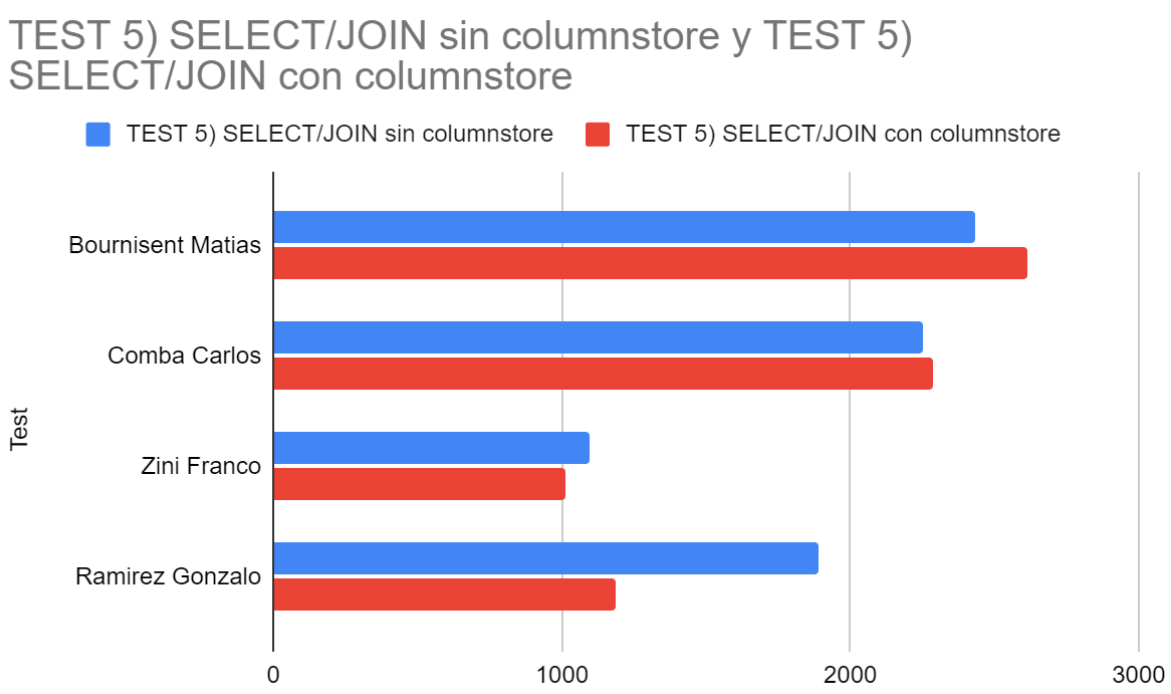
Graficamos los tiempos para cada integrante en cada test




TEST 2) AVG(importe) sin columnstore y TEST 2) AVG(importe) con columnstore
TEST 2) AVG(importe) sin columnstore
TEST 2) AVG(importe) con columnstore
Bournisent Matias
Comba Carlos
Zini Franco
Ramirez Gonzalo
Test
0
100
200
300


TEST 3) COUNT(*) sin columnstore y TEST 3) COUNT(*) con columnstore
TEST 3) COUNT(*) sin columnstore
TEST 3) COUNT(*) con columnstore
Bournisent Matias
Comba Carlos
Zini Franco
Ramirez Gonzalo
Test
0
50
100
150
200
250





Nuestros resultados muestran que la tabla con índice columnar, "gastonew", en la mayoría de los casos exhibe un mejor rendimiento en términos de tiempo de respuesta y costo en comparación con la tabla "gasto". En cada uno de los tests realizados, observamos tiempos de respuesta más rápidos y costos de consulta

sustancialmente menores en "gastonew". Estos resultados respaldan de manera concluyente la eficacia del índice columnar en la optimización del acceso a los datos. Sin embargo se observan dos casos en los cuales uno posee peor rendimiento con índice columnar, y otro en el que la mejora es sustancialmente mayor, atribuimos esto a diferencias en el hardware o configuración de cada entorno de ejecución.

Capítulo V: Conclusión

En conclusión, los índices columnares en SQL Server representan una herramienta fundamental para mejorar el rendimiento de las bases de datos. Esto se traduce en una mayor eficiencia operativa y en la capacidad de tomar decisiones más informadas. La implementación de índices columnares marca la diferencia en la experiencia de los usuarios y en el éxito de las aplicaciones y sistemas que dependen de una gestión eficaz de datos.

La estrategia de implementar índices columnares resulta ser de un valor incalculable para las organizaciones que buscan maximizar el valor de sus bases de datos. Al enfocarse en columnas específicas en lugar de en la tabla completa, los índices columnares permiten un acceso más rápido y eficiente a la información relevante. Como hemos demostrado en nuestras pruebas de rendimiento, la diferencia en tiempo y costo entre una columna con índice columnar y una columna sin él es sustancial, los índices columnares no solo aceleran las consultas, sino que también reducen los costos asociados con las operaciones de consulta, lo que tiene un impacto positivo en la eficiencia de la organización. En un mundo donde el acceso rápido y preciso a los datos es esencial, los índices columnares son una herramienta esencial.

Capítulo VI: Bibliografía

1. SQL Performance Explained" por Markus Winand
2. Fundamentos de Base de Datos" de Korth H. y Silberschatz A. (1993, McGraw-Hill, ISBN: 84-481-0079-4)
3. Microsoft. (2023). Introducción a los índices de almacén de columnas. https://learn.microsoft.com/es-es/sql/relational-databases/indexes/columnstore-indexes-overview?view=sql-server-ver16
4. blogvisionarios (2012). ColumnStore Index–Índices columnares en SQL https://blogvisionarios.com/articulos-data/columnstore-index-indices-columnares-en-sql/
5. Redgate (2019). What are Columnstore Indexes? https://www.red-gate.com/simple-talk/databases/sql-server/t-sql-programming-sql-server/what-are-columnstore-indexes/