Universidad Nacional del Nordeste Facultad de Ciencias Exactas y Naturales yAgrimensura

BASE DE DATOS I

“Optimización de consultas a través de índices”

Año 2023

Profesor: Walter Oscar Vallejos

INTEGRANTES:

Aromi, Luciano

Baez, Debora

Benitez, Mariana

Chicala Alejandro

Tabla de contenido

CAPITULO I

Introducción

En el presente trabajo de investigación, nos adentraremos en el campo de la "Optimización de consultas a través de índices" en el ámbito de la auditoría de bases de datos. Nuestro objetivo como grupo es recopilar información sólida y completa sobre este tema, con el propósito de comprenderlo a fondo y asimilar sus conceptos. Esto nos permitirá no solo adquirir el conocimiento necesario para ofrecer una presentación informativa sobre el tema, dirigida tanto a nuestros compañeros de carrera como a nuestros profesores, sino también para aplicarlo en proyectos y trabajos futuros.

La optimización de consultas a través de índices es una técnica utilizada en las bases de datos para acelerar la velocidad de las consultas. Los índices son usados para encontrar rápidamente los registros que tengan un determinado valor en alguna de sus columnas. Sin un índice, el sistema de gestión de bases de datos tiene que iniciar con el primer registro y leer a través de toda la tabla para encontrar los registros relevantes.

CAPITULO II

Marco conceptual

Las consultas basadas en índices y las consultas sin índices son fundamentalmente las mismas en términos de la sintaxis SQL que utilizas. La diferencia radica en cómo el sistema de gestión de bases de datos (DBMS) procesa estas consultas.

Cuando realizas una consulta en una tabla sin índices, el DBMS tiene que realizar una búsqueda completa de la tabla, también conocida como búsqueda secuencial. Esto significa que tienes que revisar cada fila de la tabla para encontrar las filas que cumplen con la condición de tu consulta. Esto puede ser muy lento si la tabla tiene muchas filas.

Por otro lado, cuando realizas una consulta en una tabla con un índice en la columna que estás buscando, el DBMS puede utilizar el índice para encontrar las filas que cumplen con la condición de tu consulta de manera mucho más rápida. En lugar de tener que buscar en todas las filas de la tabla, puede buscar directamente en el índice, que está diseñado para permitir búsquedas rápidas, y luego recuperar sólo las filas correspondientes de la tabla. Esto se conoce como búsqueda indexada.

Por lo tanto, la principal diferencia entre las consultas basadas en índices y las consultas sin índices es la velocidad a la que se pueden ejecutar. Las consultas basadas en índices suelen ser mucho más rápidas para las tablas grandes, pero requieren que se mantenga un índice, lo cual puede ralentizar las operaciones de escritura y ocupar espacio adicional en disco.

Tenemos como objetivo entonces explicar mediante un ejemplo practico proporcionado por la catedra el funcionamiento de este tipo de consultas y su impacto en el rendimiento.

CAPITULO III

Metodología

Para llevar a cabo este proyecto, después de recibir el tema asignado, optamos por crear un grupo de WhatsApp con todos los miembros con el propósito de mantener una comunicación efectiva. Este canal de comunicación nos permitió expresar nuestras ideas, resolver dudas y coordinar reuniones, entre otras tareas necesarias para el desarrollo del trabajo.

En cuanto a la metodología que decidimos seguir, acordamos que cada miembro del grupo se responsabilizará de recopilar información relevante sobre el tema, ya sea a través de recursos en línea, libros u otras fuentes disponibles. De esta manera, al reunirnos, todos contábamos con un conjunto de materiales que podríamos aportar al proyecto.

Durante nuestras reuniones, cada uno presentó el material que había recopilado, y en conjunto procedimos a completar cada sección de nuestro trabajo. Mientras avanzábamos en la fase de desarrollo, revisamos la información recopilada, extraemos conclusiones y determinamos qué datos eran más destacables y apropiados para ser incluidos en nuestro proyecto de investigación.

En cuanto a la aplicación de conocimientos en el ejemplo practico se realizaron las pruebas solicitadas de las cuales llevamos nota para comprender el funcionamiento del motor de base de datos.

CAPITULO IV

¿Cómo funcionan los índices?

Los índices funcionan de manera similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en la base de datos. Para buscar un elemento que esté indexado, sólo hay que buscar en el índice dicho elemento para, una vez encontrado, devolver un registro que se encuentre en la posición marcada por el índice.

Tipos de Índices

Existen varios tipos de índices que se pueden utilizar en función de nuestras necesidades. Los más comunes son:

Índice agrupado (CLUSTERED)

> Este índice determina el orden físico de los registros en la tabla. **Cada tabla puede tener solo un índice agrupado** porque define la estructura de almacenamiento de la tabla misma. Los datos de la tabla se almacenan en el orden especificado por el índice agrupado. Por lo tanto, estos son útiles para consultas que recuperan un rango de datos ordenados de manera específica.

Índice no agrupado (Nonclustered):

Los índices no agrupados son adicionales a la tabla principal y no afectan el orden físico de los registros en la tabla. Puedes tener varios índices no agrupados en una tabla. Estos índices contienen copias de una o varias columnas de la tabla con un puntero a la fila correspondiente. Son adecuados para acelerar consultas de búsqueda y facilitar la selección de datos basados en valores en las columnas indexadas.

Índice Compuesto

Un índice compuesto se crea en múltiples columnas. Puede ser útil para mejorar el rendimiento de las consultas que involucran una combinación de valores de esas columnas.

Otros índices comunes:

Índices únicos: Este tipo de índice garantiza que los valores en la columna indexada sean únicos en la tabla. Se utiliza para hacer cumplir restricciones de unicidad en una columna específica.

Índices textuales: Se utilizan para realizar consultas que involucran la extracción de información textual de la base de datos.

Ventajas y Desventajas

Ventajas

De ser utilizados correctamente, los indices pueden mejorar el rendimiento de la siguiente manera:

- Rapidez: Los índices permiten una mayor rapidez en la ejecución de las consultas y recuperación de datos.
- Eficiencia Los índices mejoran la eficiencia al permitir un acceso más rápido a las filas de una tabla.

Desventajas

Los indices deben ser equilibrados y ser creados a partir de un analissis ded las consultas mas frecuentes, de lo contrario pueden generar:

- Mayor complejidad: El utilizar y administrar adecuadamente una gran cantidad de base de datos distribuidos en cientos de nodos es una tarea muy compleja.
- Consumo de recursos: Los índices también consumen recursos, tanto de almacenamiento como de procesamiento, sobre todo utilizados de manera incorrecta.

Elección de columnas a indexar

La elección de las columnas a indexar en una base de datos es un aspecto crítico para lograr un rendimiento eficiente en las consultas. Indexar las columnas adecuadas puede acelerar la recuperación de datos, pero indexar en exceso o de manera inapropiada puede tener efectos negativos en el rendimiento general de la base de datos. Aquí hay algunas pautas para elegir las columnas a indexar:

1. **Columnas de Frecuente Búsqueda**:
    - Las columnas que se utilizan con frecuencia en cláusulas WHERE en consultas de búsqueda son candidatas ideales para la indexación. Esto incluye columnas que se utilizan en condiciones de igualdad o comparaciones de rango.
2. **Columnas de Unicidad**:
    - Las columnas que deben contener valores únicos, como identificadores, son buenos candidatos para índices únicos. Esto garantiza que no haya duplicados en la columna y acelera la búsqueda por valor exacto.
3. **Columnas de Join**:
    - Si realizas operaciones de JOIN con frecuencia en tablas relacionadas, indexar las columnas involucradas en las condiciones de JOIN mejora significativamente el rendimiento.
4. **Columnas de Ordenamiento**:
    - Columnas utilizadas en operaciones de ordenamiento (por ejemplo, en cláusulas ORDER BY) deben considerarse para la indexación, ya que acelerarán estas operaciones.
5. **Columnas Filtradas**:
    - En algunos casos, es útil crear índices en columnas calculadas o filtradas. Esto permite optimizar consultas basadas en cálculos específicos.
6. **Columnas de Texto Completo**:
    - Si tienes columnas de texto en las que se realizan búsquedas de texto completo, considera la creación de índices de texto completo para mejorar la velocidad de estas consultas.
7. **Columnas de Filtros Comunes**:
    - Si tienes columnas que se utilizan comúnmente para filtrar datos, es importante indexarlas para acelerar la recuperación de registros.
8. **Columnas con Alto Cardenalidad**:
    - Las columnas con un alto número de valores distintos (alta cardinalidad) suelen beneficiarse de la indexación, ya que reducen la cantidad de registros que deben ser explorados.

9. **Monitoreo y Ajuste**:
    - Realiza un seguimiento del rendimiento de las consultas y ajusta los índices según sea necesario. La creación de índices innecesarios puede ralentizar las operaciones de escritura.

Es importante recordar que no es necesario (y, a veces, es contraproducente) indexar todas las columnas en una tabla. La elección de las columnas a indexar **debe basarse en las consultas y patrones de acceso a los datos reales de tu aplicación**. Además, el mantenimiento de índices es esencial, ya que los índices **deben actualizarse cuando se realizan cambios en los datos**. El equilibrio entre el rendimiento de las consultas y el costo de mantenimiento de los índices es fundamental.

Funcionamiento del motor de base de datos

En esta sección, exploraremos en detalle cómo funciona el motor de base de datos al utilizar índices agrupados y no agrupados en el contexto de consultas SQL.

Índices No Agrupados (Non-Clustered Index)

1. **Búsqueda en el Índice No Agrupado**:
    - En el caso de un índice no agrupado, el motor de base de datos accede al índice no agrupado correspondiente.
    - Busca directamente la clave de búsqueda en el índice no agrupado.
2. **Ubicación de la Clave de Búsqueda**:
    - Una vez que se encuentra la clave de búsqueda en el índice no agrupado, el motor obtiene uno o más identificadores de filas (punteros) que corresponden a esa clave en la tabla principal.
3. **Recuperación de Datos Parciales**:
    - Con los identificadores de filas obtenidos del índice no agrupado, el motor accede a la tabla principal.
    - Recupera solo las columnas necesarias que cumplen con los criterios de la consulta, en lugar de todas las columnas de la tabla.
4. **Entrega de Resultados**:
    - Los registros parciales que cumplen con los criterios de la consulta se entregan como resultado de la consulta.

Índices Agrupados (Clustered Index)

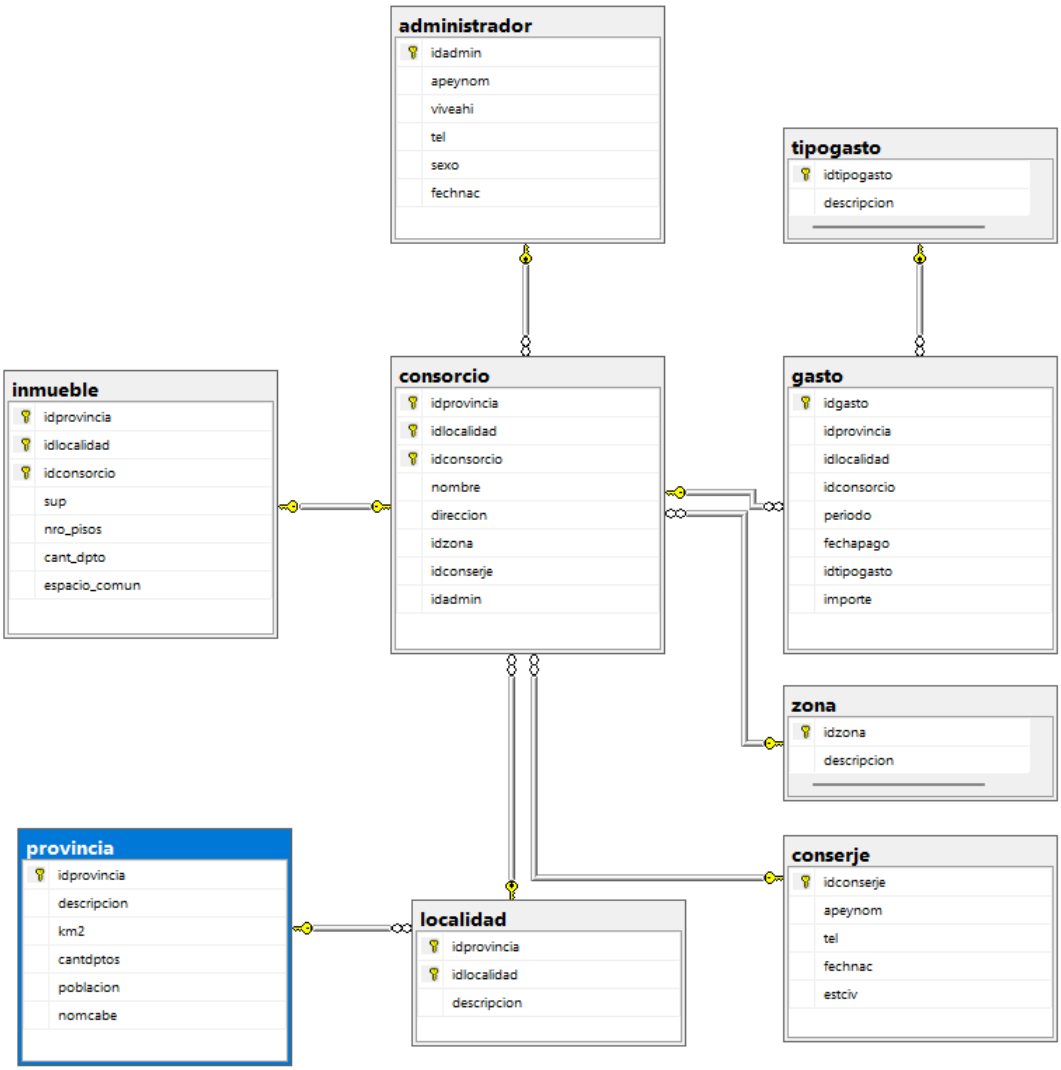
1. **Búsqueda en el Índice Agrupado**:
    - Cuando se ejecuta una consulta, el motor de base de datos accede al índice agrupado correspondiente.
    - Busca directamente la clave de búsqueda (por ejemplo, un valor de columna) en el índice agrupado.

2. **Ubicación del Registro**:
    - Una vez que se encuentra la clave de búsqueda en el índice agrupado, el motor obtiene el identificador del registro en la tabla principal.
    - Este identificador generalmente es un puntero que apunta al registro físico en la tabla.
3. **Recuperación de Datos Completos**:
    - Con el identificador del registro, el motor accede a la tabla principal y recupera el registro completo que cumple con los criterios de la consulta.
    - Esto incluye todas las columnas del registro.
4. **Entrega de Resultados**:
    - Los registros que cumplen con los criterios de la consulta se entregan como resultado de la consulta.

Caso de estudio

En este informe, presentamos un caso de estudio de una base de datos diseñada para gestionar información relacionada con inmuebles, consorcios y gastos asociados. El objetivo principal de esta base de datos es permitir un seguimiento eficiente de los datos relacionados con propiedades inmobiliarias, facilitar la gestión de consorcios y brindar información sobre los gastos asociados.

Diagrama de Entidad-Relación (DER)



Modelo de Datos (Diseño Relacional):

Tabla provincia:

- Descripción: Almacena información sobre las provincias.
- Atributos:
  - `idprovincia` (Clave primaria)
  - `descripcion`
  - `km2`
  - `cantdptos`
  - `poblacion`
  - `nomcabe`

Tabla localidad:

- Descripción: Almacena información sobre las localidades y su relación con las provincias.

- Atributos:
  - `idprovincia` (Clave foránea a `provincia`)
  - `idlocalidad` (Clave primaria)
  - `descripcion`

Tabla zona:

- Descripción: Almacena información sobre las zonas.
- Atributos:
  - `idzona` (Clave primaria)
  - `descripcion`

Tabla conserje:

- Descripción: Almacena información sobre los conserjes.
- Atributos:
  - `idconserje` (Clave primaria)
  - `apeynom`
  - `tel`
  - `fechnac`
  - `estciv`

Tabla administrador:

- Descripción: Almacena información sobre los administradores y su relación con los consorcios.
- Atributos:
  - `idadmin` (Clave primaria)
  - `apeynom`
  - `viveahi`
  - `tel`
  - `sexo`
  - `fechnac`

Tabla tipogasto:

- Descripción: Almacena información sobre los tipos de gastos.
- Atributos:
  - `idtipogasto` (Clave primaria)
  - `descripcion`

Tabla consorcio:

- Descripción: Almacena información sobre los consorcios y su relación con provincias, localidades, zonas, conserjes y administradores.
- Atributos:
  - `idprovincia` (Clave foránea a `provincia`)
  - `idlocalidad` (Clave foránea a `localidad`)
  - `idconsorcio` (Clave primaria)
  - `nombre`

- `direccion`
- `idzona` (Clave foránea a `zona`)
- `idconserje` (Clave foránea a `conserje`)
- `idadmin` (Clave foránea a `administrador`)

Tabla gasto:

- Descripción: Almacena información sobre los gastos y su relación con consorcios y tipos de gastos.
- Atributos:
  - `idgasto` (Clave primaria)
  - `idprovincia` (Clave foránea a `provincia`)
  - `idlocalidad` (Clave foránea a `localidad`)
  - `idconsorcio` (Clave foránea a `consorcio`)
  - `periodo`
  - `fechapago`
  - `idtipogasto` (Clave foránea a `tipogasto`)
  - `importe`

Diccionario de Datos:

A continuación, se presenta un diccionario de datos para algunas de las columnas clave en las tablas del modelo de datos:

Tabla provincia:

- `idprovincia` (INT):
  - Restricciones y Validaciones: Clave primaria (PK)
  - Descripción: Identificador único de la provincia.
- `descripcion` (VARCHAR(50)):
  - Descripción: Nombre o descripción de la provincia.
- `km2` (INT):
  - Descripción: Área en kilómetros cuadrados de la provincia.

Tabla localidad:

- `idprovincia` (INT):
  - Restricciones y Validaciones: Clave foránea (FK) referente a `provincia.idprovincia`
  - Descripción: ID de la provincia a la que pertenece la localidad.

- `idlocalidad` (INT):
  - Restricciones y Validaciones**: Clave primaria (PK)
  - Descripción: Identificador único de la localidad.

- `descripcion` (VARCHAR(50)):
  - Descripción: Nombre o descripción de la localidad.

Tabla zona:

- `idzona` (INT):

  - Restricciones y Validaciones: Clave primaria (PK)
  - Descripción: Identificador único de la zona.

- `descripcion` (VARCHAR(50)):
  - Descripción: Nombre o descripción de la zona.

Tabla conserje:
- `idconserje` (INT):
  - Restricciones y Validaciones: Clave primaria (PK)
  - Descripción: Identificador único del conserje.

- `apeynom` (VARCHAR(50)):
  - Descripción: Apellido y nombre del conserje.

- `tel` (VARCHAR(20)):
  - **Descripción**: Número de teléfono del conserje.

- `fechnac` (DATETIME):
  - Descripción: Fecha de nacimiento del conserje.

- `estciv` (CHAR(1)):
  - Restricciones y Validaciones: Valores posibles: 'S' (Soltero), 'C' (Casado), 'D' (Divorciado), 'O' (Otro).
  - Descripción: Estado civil del conserje.

Tabla administrador:
- `idadmin` (INT):
  - Restricciones y Validaciones: Clave primaria (PK)
  - Descripción: Identificador único del administrador.

- `apeynom` (VARCHAR(50)):
  - Descripción: Apellido y nombre del administrador.

- `viveahi` (CHAR(1)):
  - Restricciones y Validaciones: Valores posibles: 'S' (Sí), 'N' (No).
  - Descripción: Indica si el administrador vive en la misma ubicación que el consorcio.

- `tel` (VARCHAR(20)):
  - Descripción: Número de teléfono del administrador.

- `sexo` (CHAR(1)):
  - Restricciones y Validaciones: Valores posibles: 'F' (Femenino), 'M' (Masculino).
  - Descripción: Sexo del administrador.

- `fechnac` (DATETIME):
  - Descripción: Fecha de nacimiento del administrador.

Tabla tipogasto:

- `idtipogasto` (INT):
  - Restricciones y Validaciones: Clave primaria (PK)
  - Descripción: Identificador único del tipo de gasto.

- `descripcion` (VARCHAR(50)):
  - Descripción: Descripción del tipo de gasto.

Tabla `consorcio`:

- **`idprovincia` (INT)**:
  - **Restricciones y Validaciones**: Clave foránea (FK) referente a `provincia.idprovincia`.
  - **Descripción**: ID de la provincia a la que pertenece el consorcio.

- **`idlocalidad` (INT)**:
  - **Restricciones y Validaciones**: Clave foránea (FK) referente a `localidad.idlocalidad`.
  - **Descripción**: ID de la localidad a la que pertenece el consorcio.

- `idconsorcio` (INT):
  - Restricciones y Validaciones: Clave primaria (PK).
  - Descripción: Identificador único del consorcio.

- `nombre` (VARCHAR(50)):
  - **Descripción**: Nombre del consorcio.

- `direccion` (VARCHAR(250)):
  - Descripción: Dirección del consorcio.

- `idzona` (INT):
  - Restricciones y Validaciones: Clave foránea (FK) referente a `zona.idzona`.
  - Descripción: ID de la zona a la que pertenece el consorcio.

- `idconserje` (INT):
  - Restricciones y Validaciones: Clave foránea (FK) referente a `conserje.idconserje`.
  - Descripción: ID del conserje asignado al consorcio.

- `idadmin` (INT):
  - Restricciones y Validaciones: Clave foránea (FK) referente a `administrador.idadmin`.
  - Descripción: ID del administrador del consorcio.

Tabla gasto:

- `idgasto` (INT):
  - Restricciones y Validaciones: Clave primaria (PK).
  - Descripción: Identificador único del gasto.

- `idprovincia` (INT):
  - Restricciones y Validaciones: Clave foránea (FK) referente a `provincia.idprovincia`.
  - Descripción: ID de la provincia a la que pertenece el gasto.

- `idlocalidad` (INT):
  - Restricciones y Validaciones: Clave foránea (FK) referente a `localidad.idlocalidad`.
  - Descripción: ID de la localidad a la que pertenece el gasto.

- `idconsorcio` (INT):
  - Restricciones y Validaciones: Clave foránea (FK) referente a `consorcio.idconsorcio`.
  - Descripción: ID del consorcio al que está asociado el gasto.
- `periodo` (INT):
  - Descripción: Período al que corresponde el gasto.

- `fechapago` (DATETIME):
  - Descripción: Fecha en la que se realizó el pago del gasto.

- `idtipogasto` (INT):
  - Restricciones y Validaciones: Clave foránea (FK) referente a `tipogasto.idtipogasto`.
  - Descripción: ID del tipo de gasto.
- `importe` (DECIMAL(8,2):
  - Descripción: Importe del gasto.

Ejemplo practico

Para demostrar el impacto de los índices aplicamos las modificaciones que se nos requerían

Primera consulta sin índices

Realizamos nuestra primera consulta sin modificaciones sobre la tabla “gasto” sobre un lote de 1006046 registros, 8000 de ellos fueron cargados a partir del lote de datos proporcionado por los docentes, mientras que los demás fueron a través del siguiente script.

```
--Carga masiva de datos
INSERT INTO gasto (idprovincia, idlocalidad, idconsorcio, periodo,
fechapago, idtipogasto, importe)
SELECT TOP 100000
    24,4,1,8,'20170810',3,60321.49
FROM sys.objects a
CROSS JOIN sys.objects b;

SELECT count(*) FROM [dbo].[gasto];
```

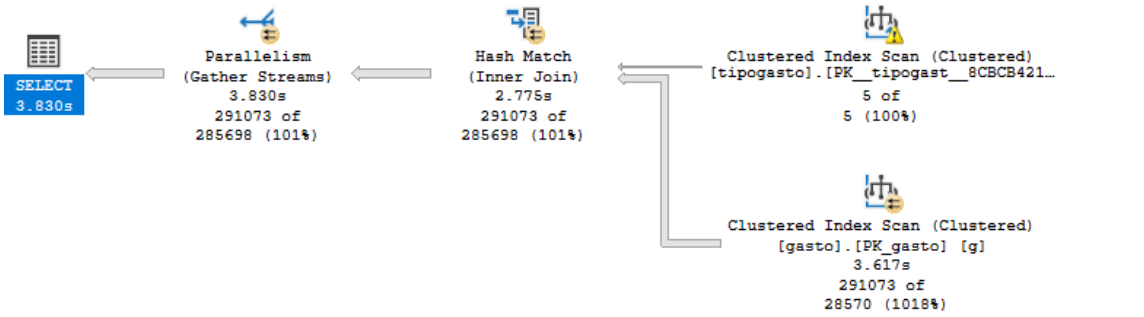
Los datos a insertar fueron alternando por cada ejecución.

Consulta

```
--Consulta: Gastos del periodo 8

SELECT g.idgasto, g.periodo, g.fechapago, t.descripcion
FROM gasto g
INNER JOIN tipogasto t ON g.idtipogasto = t.idtipogasto
WHERE g.periodo = 8;
```

Plan de ejecución



Tiempo promedio de ejecución: 3.700 ms

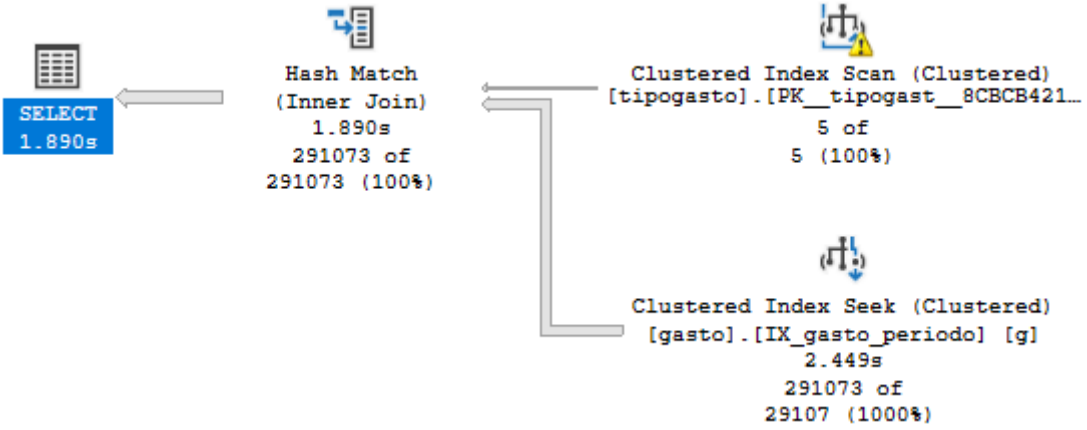
Creación del primer índice

Creamos un primer índice agrupado en el campo periodo. Para ello fue necesario primero hacer modificaciones en la tabla ya que sqlServer ya interpretaba a la clave primaria de "gasto" como un índice agrupado**. Es importante que esta modificación se haga luego de la carga masiva para evitar problemas en la inserción**

```
--SqlServer toma la PK de gasto como indice CLUSTERED, asique para crear el
solicitado en periodo debemos primero eliminar el actual
ALTER TABLE gasto
DROP CONSTRAINT PK_gasto;

--Creamos un nuevo indice CLUSTERED en periodo
CREATE CLUSTERED INDEX IX_gasto_periodo
ON gasto (periodo);
```

Plan de ejecución de la nueva consulta



Tiempo promedio de ejecución: 2000 ms

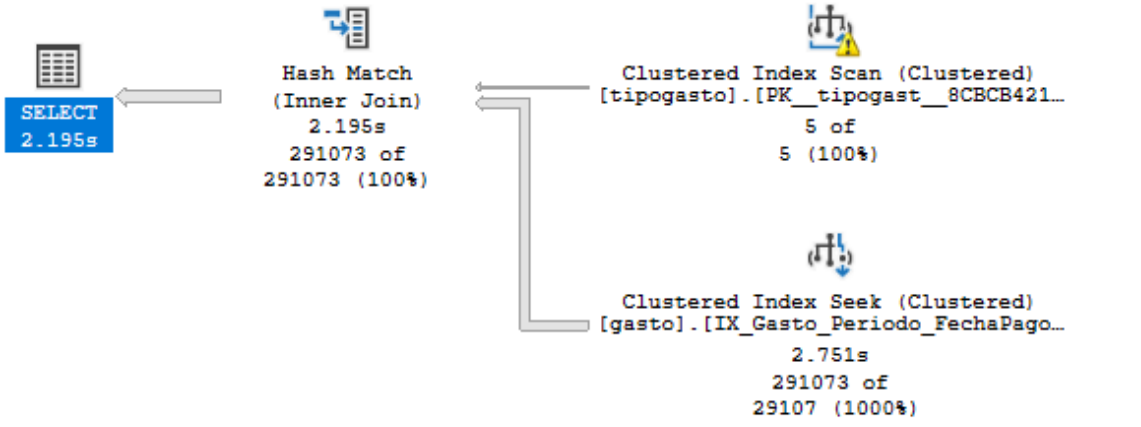
Creación de un tercer índice

Por último, creamos un nuevo índice agrupado en periodo, pero esta vez incluyendo dos columnas más, fecha de pago y tipo de gasto.

```
-- Elimina el índice agrupado anterior
DROP INDEX IX_gasto_periodo ON gasto;

-- Crea un nuevo índice agrupado en periodo, fechapago e idtipogasto
CREATE CLUSTERED INDEX IX_Gasto_Periodo_FechaPago_TipoGasto
ON gasto (periodo, fechapago, idtipogasto);
```

Plan de ejecución:



Tiempo promedio de ejecución: 2000 ms

Notamos entonces que bajo esa consulta el índice no afectaba significativamente a la optimización. Si tenemos en cuenta ese tipo de consulta, un índice agrupado en periodo bastaba para afectar al rendimiento.

CAPITULO V

Conclusiones

En conclusión, los índices desempeñan un papel fundamental en la optimización de consultas en bases de datos, incluyendo SQL Server. Aquí hay algunas conclusiones clave acerca de los índices:

1. **Mejora del Rendimiento**: Los índices aceleran la recuperación de datos al proporcionar un acceso más rápido a las filas de una tabla. Esto es especialmente beneficioso en tablas grandes.
2. **Elección de Tipos de Índices**: Es importante seleccionar el tipo de índice adecuado para una tabla y una consulta específica. Los tipos comunes de índices incluyen los índices agrupados y no agrupados.
3. **Índices Agrupados vs. No Agrupados**: Los índices agrupados determinan el orden físico de los datos en la tabla, mientras que los no agrupados son estructuras de datos adicionales que almacenan el mapeo de valores a filas.
4. **Columnas a Indexar**: Debes considerar cuidadosamente qué columnas indexar. Indexar columnas utilizadas con frecuencia en cláusulas WHERE y JOIN puede mejorar significativamente el rendimiento.
5. **Costo de Mantenimiento**: Los índices tienen un costo de mantenimiento, ya que deben actualizarse con cada inserción, actualización o eliminación de filas. Demasiados índices innecesarios pueden ralentizar la carga de datos.
6. **Equilibrio**: La creación de índices es un proceso de equilibrio. Debes encontrar el equilibrio adecuado entre el número de índices y la necesidad de mejorar el rendimiento.

En resumen, los índices son una herramienta valiosa para acelerar las consultas, pero deben utilizarse con cuidado. La elección de cuándo y qué indexar depende de las necesidades específicas de las consultas y del entorno de la base de datos. El monitoreo, análisis y ajuste periódico de los índices son prácticas clave para mantener un rendimiento óptimo en una base de datos.

CAPITULO VI

Bibliografía

Documentación de Microsoft SQL Server:

- En el texto: (Microsoft, 2023)
- En la lista de referencias:
    - Microsoft. (2023). Documentación de Microsoft SQL Server. https://docs.microsoft.com/en-us/sql/sql-server/

SQL Server Index Design Guide:

- En el texto: (Microsoft, 2023a)
- En la lista de referencias:
    - Microsoft. (2023). SQL Server Index Design Guide. https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide