Optimización de consultas a través de índices

¿Qué es un Índice?

Un índice es simplemente un “atajo” para encontrar información dentro de un gran conjunto de datos, por lo general, dichos índices se referencian con caracteres simples o números. El ejemplo perfecto o más cotidiano en un Índice alfabético al final de un libro de texto, en vez de buscar página por página alguna palabra clave de la cual queremos saber. Es en este momento donde tenemos dos posibles caminos para el hallazgo de la misma 1. podríamos buscar página por página aquello que queremos saber utilizando una palabra clave como referencia. 2. Utilizamos el índice alfabético para encontrar dicha palabra y luego dirigirnos a la página exacta sin muchas complicaciones.

Ahora aplicado más al contexto en donde nos encontramos, es decir, índices para base de datos, es una estructura de datos especiales, separada de una tabla principal, que almacena una copia de una o más columnas de forma ordenada.
Específicamente sería un **puntero**(como el número de página) que le dice al motor de base de datos donde se encuentra la fila completa.

Índice Agrupado(Clustered Index)

Es aquel que define el orden físico en que las filas de datos se almacenan en el disco. No es un objeto separado de la tabla, la tabla misma se convierte en el índice. Debido a que los datos solo pueden estar almacenados físicamente en un único orden, una tabla sólo puede tener un índice agrupado.
La misma posee una estructura de (B-Tree) o Árbol-B:

- Los nodos raíz e intermedios del árbol contienen valores de la clave del índice que actúan como "señales" para guiar la búsqueda.

- El nivel más bajo del árbol, las hojas (leaf nodes), no contienen punteros. Las hojas del índice agrupado son las páginas de datos reales de la tabla.

- Esto significa que los datos de la tabla están organizados y ordenados secuencialmente en el disco según la clave del índice agrupado (por ejemplo, `id_paciente`). Cuando buscás por esa clave, la base de datos va directamente a la ubicación física del dato.

Ventajas

- [ ] **Rendimiento de Lectura Insuperable (en Búsquedas por Clave):** Para búsquedas exactas (`WHERE id_paciente = 123`) o, especialmente, búsquedas por rango (`WHERE id_paciente BETWEEN 100 AND 200`), es el más rápido. ¿Por qué? Porque los datos están físicamente contiguos en el disco. SQL Server no "salta", simplemente "sigue leyendo" el disco secuencialmente.
- [ ] **Sin Paso Adicional (No "Key Lookup"):** Cuando la búsqueda encuentra la fila en el índice, ha encontrado la fila de datos completa (nombre, apellido, DNI, todo). No hay que hacer un "salto" extra para buscar el resto de los datos, porque el índice *es* los datos.
- [ ] **Almacenamiento Eficiente (para el índice):** No crea una copia de los datos, por lo que el índice en sí mismo no duplica el almacenamiento (aunque la estructura B-Tree tiene un pequeño *overhead*).

Limitaciones

- **Solo UNO por Tabla:** Esta es su mayor limitación. Debes elegir *muy* cuidadosamente qué columna será tu clave agrupada, ya que solo puedes ordenar físicamente la tabla de una manera. (En tu proyecto, `id_paciente` es la elección obvia para `Paciente`, pero ¿qué pasaría si el 99% de las búsquedas fueran por `dni`? Sería un dilema de diseño).

  Costoso en Escrituras (`INSERT`, `UPDATE`): Esta es su mayor desventaja.

- **`INSERT`:** Cuando insertamos una fila nueva, SQL Server debe colocarla **físicamente en su lugar ordenado**. Si la página de datos donde debe ir está llena, ocurre una **"División de Página" (Page Split)**: SQL Server debe crear una nueva página, mover la mitad de los datos de la página vieja a la nueva y luego insertar la fila. Esto es *muy* costoso y genera fragmentación.

- **`UPDATE`:** Si actualizas el valor de la clave agrupada (ej. cambiar el `id_paciente` de 1 a 1000), SQL Server debe **borrar físicamente la fila** de su ubicación original y **moverla por completo** a la nueva ubicación. Es una operación extremadamente lenta (por eso casi siempre se usan claves que no cambian, como un `IDENTITY`).

Índice No Agrupado (Non-Clustered Index)

es una estructura de datos completamente separada de la tabla. Es un "mini-libro" auxiliar que existe independientemente del orden físico de los datos de la tabla. Cómo son estructuras separadas, una tabla puede tener muchos índices no agrupados.
Como los Agrupados también posee una estructura de B-Tree:

- La gran diferencia está en el nivel de las hojas (leaf nodes). Las hojas de un índice no agrupado **NO** contienen la fila de datos completa.

    En su lugar, las hojas contienen dos cosas:

1. El valor de la clave del índice (ej. el `dni` 45.527.225).
2. Un **"Localizador de Fila" (Row Locator)**. Este localizador es un puntero que le dice a la base de datos dónde encontrar la fila de datos completa. Este puntero es, por lo general, la clave del índice agrupado de la tabla (ej. el `id_paciente = 1`).

Ventajas

- **Múltiples Índices por Tabla:** Esta es su mayor fortaleza. Puedes tener muchos. En tu proyecto, la tabla `Paciente` puede tener un índice agrupado en `id_paciente` **y además** un índice no agrupado en `dni`, otro en `(apellido, nombre)`, etc. Te permite optimizar *muchos tipos diferentes* de consultas.

- **Escrituras (`INSERT`) más Rápidas (que el Clustered):** Un `INSERT` sigue siendo trabajo (hay que añadir la entrada al índice), pero generalmente es más rápido que en un índice agrupado. La fila de datos principal simplemente se añade al final de la tabla (si la tabla es un "Heap") o en su lugar del Clustered Index, y el índice no agrupado solo añade una pequeña entrada de puntero en su propia estructura.

- **Flexibilidad:** Son la herramienta principal para el "tuning" (afinamiento) de consultas.

Limitaciones

    - **Costo de "Key Lookup" (Búsqueda de Clave):** Esta es su principal desventaja. Si ejecutas `SELECT * FROM Paciente WHERE dni = 45527225;`:

1. SQL Server busca el `dni` en el índice no agrupado (rápido).
2. Obtiene el puntero (el `id_paciente`).
3. Debe hacer un **segundo salto** (el "Key Lookup") al índice agrupado (la tabla) para buscar por `id_paciente` y obtener el resto de las columnas (`nombre`, `direccion`, etc.).

- Este "doble salto" tiene un costo. Si la consulta devuelve *muchas* filas, hacer miles de "Key Lookups" puede ser más lento que un "Full Table Scan".

- **Costo de Almacenamiento:** Cada índice no agrupado que creas es un **objeto nuevo** que ocupa espacio adicional en tu base de datos.

- **Sobrecarga de Escritura (Overhead):** Este es el "costo oculto". Si tu tabla `Paciente` tiene 1 índice agrupado y 5 no agrupados, un solo `INSERT` provoca **6**

operaciones de escritura (una en la tabla y una en cada índice). Esto se conoce como **"Sobre-indexación" (Over-indexing)** y puede hacer que tus `INSERT`, `UPDATE` y `DELETE` sean *extremadamente* lentos.