Apunte - Semana 11 - 7/10/2022

Max Richard Lee Chung - 2019185076

> La entrega del Proyecto 1 se traslada para semana 13, martes 18, a la misma hora.

Índices

Los índices son diferentes en cada sistema, en donde se represetan como árboles binarios y buscan mejorar el rendimiento reduciendo la duración de búsqueda.

Clustered Index

Se crea un único indice dentro del archivo sin importar el órden y dura menos buscando los datos.

Se debe utilizar dependiendo de la carga de trabajo (workload) del rendimiento, como por ejemplo:

```
SELECT nombre, edad, fechaN FROM Persona
WHERE edad >= 30 AND provincia = 'x'
```

donde la sentencia tiende a ser muy rápido en modo clustered, sin embargo, si le agregamos a la consulta

```
ORDER BY provincia
```

aumentará mucho el costo del rendimiento por el órden de las columnas a consultar. Si se hace un "ORDER BY" con columnas fuera de la tabla (INNER JOIN) es lo peor porque hay que hacer un ordenamiento secuencial para revisar las dos tablas.

Non-clustered Index

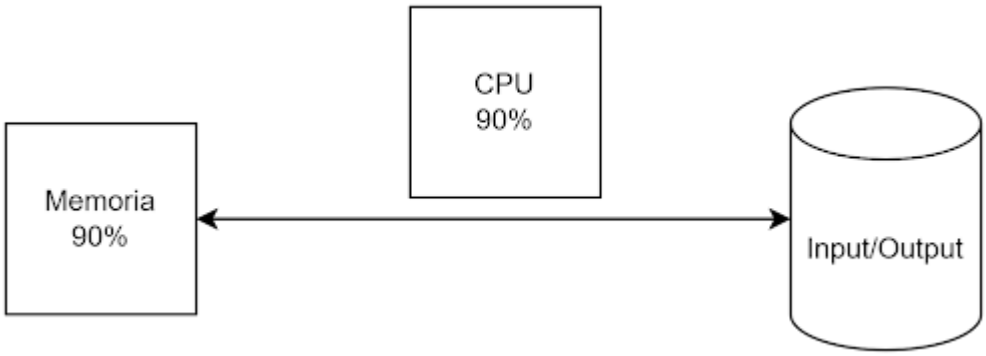
Son archivos separados con datos, de los cuales tienen punteros con el siguiente archivo con la información (archivo apunta otro archivo). Funciona como un heap de datos, es decir, entra datos y se pega al final. Normalmente, se tienen que buscar los datos de forma aleatoria, sin embargo hay formas para evitarlo, como por ejemplo, cuando se crea un índice no clúster se usa la palabra reservada INCLUDE como se puede observar en el siguiente ejemplo:

```
SELECT col1, col2, col3 FROM table WHERE edad >= 'x'
CREATE INDEX ON edad INCLUDE col1, col2, col3
```

si crea el indice en edad, se guardan las columnas col1, col2 y col3 para devolver el dato de forma rápida si es una consulta común.

Esto puede afectar a la lectura y escritura, ya que se obliga a escribir en tres archivos diferentes, guardarlos y actualizarlos.

> Lo ideal con la lectura de datos es insertar la información en memoria principal, con el fin de nunca entrar a disco (consume tiempo tiempo de ejecución)
>
> > Memory footprint: Trata de tener los datos en disco y qué tanta memoria consume la base de datos.



> El consumo alto de memoria es normal, sin embargo, un alto consumo del disco no es normal (excepto en el uso de diseño y operaciones complejas) de I/O, el cual puede ocasionar un "swapping" por estar moviendo constantemente datos entre memoria y disco por las constantes peticiones de la base de datos (las bases de datos clouding no tienen dicho término).

Se pueden tener n cantidades de índices clúster (mal diseño pero se puede realizar) con "INCLUDES", como por ejemplo, considere un Index 1 con colA, colB y colC y un Index 2 con colB, colC y colD como se puede observar a continuación.

```
SELECT * FROM table WHERE A = 'x', C = 'y', D = 'z'
```

Dicho ejemplo debe de buscar en 2 índices para sacar los registros y es mejor tener un único índice para jalar los datos en una búsqueda.

Hash Index

Lo más usado en sistemas no distribuidos.

Utiliza una función para crear índices a partir de un valor otorgado para crear el identificador en la tabla y distribuye los datos en las bases de datos.

Se puede añadir otro índice para mejorar la organización de la columna duplicada.

Expression Index

Supongamos:

```
SELECT * FROM table WHERE year(fechaN) >= 1985 AND month(fechaN) = 12
```

Cada vez que se corre, se ejecutan funciones que suman milisegundos (ms) aunque tenga una fechaN-index. Aumenta el rendimiento en lectura pero decae en escritura. Por lo tanto, hay que hacer un índice de la función para que los datos estén pre-cargados.

Partial Index

No tiene índices para todos los datos dado que se excluyen algunos datos.

```
CREATE index ON fechaN
WHERE year(fechaN) > 1985
```

Entre más pequeño, más chance hay de meterlo a memoria y evitar el "swapping" o bajar a disco. Hay que tener cuidado con las consultas y conocer los casos de usos.

> Dato curioso: Antes habían problemas con respecto al "SQL injection" porque las connsultas se realizaban desde la interfaz gráfica del usuario, por lo que los usuarios podían solicitar información extra.

Inverted Index

Selecciona documentos con textos pequeños para partir en pedazos, se parados por espacios o signos para conformar dos tipos de estructuras:

1. Dictionary of terms (contiene las palabras de uno o más archivos ordenados alfabéticamente)
2. Posting list (contiene el archivo donde presenta la palabra)

Realiza un análisis para tokenizar las palabras. Al consultar una palabra, se busca dentro del diccionario y luego devuelve los archivos de la lista para mostrarlos. Se pueden asignar relevancia a una palabra en una búsqueda dependiendo del uso y popularidad, como por ejemplo "mae" en España tiene poca relevancia en comparación de Costa Rica. Además, los motores de búsqueda también reciben retroalimentación para mejorar la relevancia.

Tiene tres tipos de filtros:

- Character filter: Elimina caracteres o signos para dar las palabras limpias
- Token analizer y filter: Eliminan las palabras muy repetidas

Steaming

- Dentro de motores de búsqueda SQL.
- Obtiene la raíz de una palabra para eliminar palabras repetidas.
- Guarda la raíz de la palabra para buscar la palabra y devolverla.
- Genera un árbol menos profundo.

Stop words

Añade palabras al token filter para eliminarlas y reducir la lista.