Base de Datos 2

Proyecto 1

Organización de Archivo

Cordova Amaya, Efrain
Soto Aguirre, David
Rios Vasquez, Paul

Índice general

1. Introducción

Nosotros buscamos implementar las técnicas de organización sobre una base de datos. Para poder realizar estas técnicas, nosotros elegimos una base datos que nos muestra el índice de multimillonarios de Bloomberg, la cual se divide en los campos: “Rank”, ”Name”, ”Total Net Worth”, “$ Last Change”, “$ YTD Change”, “Country” e “Industry”. Se espera que el proyecto pueda ofrecer dos de las técnicas de organización sugeridas para estos registros.

Para poder continuar, tenemos que tener claro las técnicas de organización que usaremos, para poder implementarlo con las operaciones de inserción, eliminación y búsqueda, a continuación presentaremos las que usaremos.

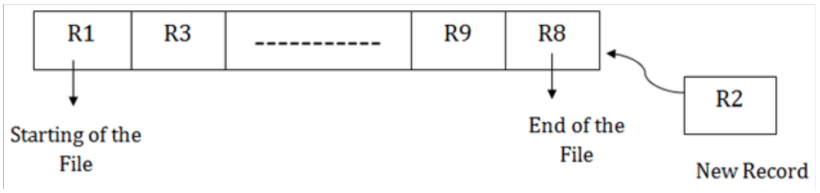
2. Técnicas

2.1. Métodos de organización

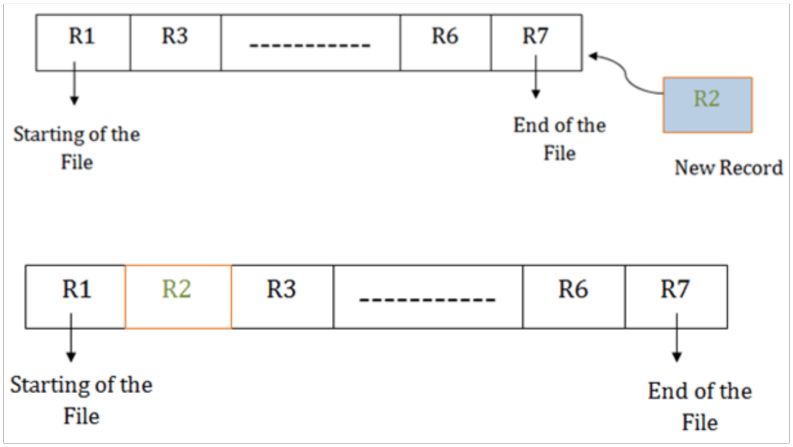
Sequential File

Los archivos se almacenan secuencialmente, se puede implementar de dos formas.

- **Método de archivo de pila:** En este método, almacenamos el registro en una secuencia, uno tras otro. En caso de actualizar o borrar algún registro, el registro se buscará en los bloques de memoria. Cuando se encuentre, se marcará para eliminarlo y se insertará el nuevo registro.

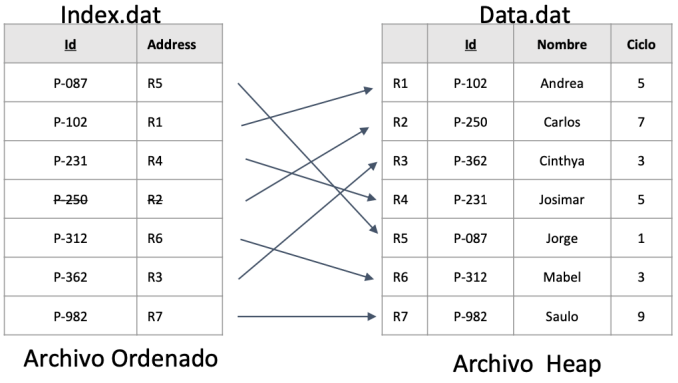


- **Método de archivo ordenado:** En este método, el nuevo registro siempre se inserta al final del archivo y luego ordenará la secuencia en orden ascendente o descendente. En el caso de modificación de algún registro, actualizará el registro y luego ordenará el archivo, y por último, el registro actualizado se colocará en el lugar correcto.



ISAM(Indexed Sequential Access Method)

El método ISAM es una organización avanzada de archivos secuenciales. En este método, los registros se almacenan en el archivo utilizando la clave principal. Se genera un valor de índice para cada clave principal y se asigna al registro. Este índice contiene la dirección del registro en el archivo.



En el presente informe, detallaremos cómo implementamos los métodos de organización ya previamente mencionados, junto con sus respectivas funciones.

2.2. Programa

Antes de aplicar cualquier método de organización, debemos definir los datos que utilizaremos para poder acceder y alterar el archivo que deseamos.

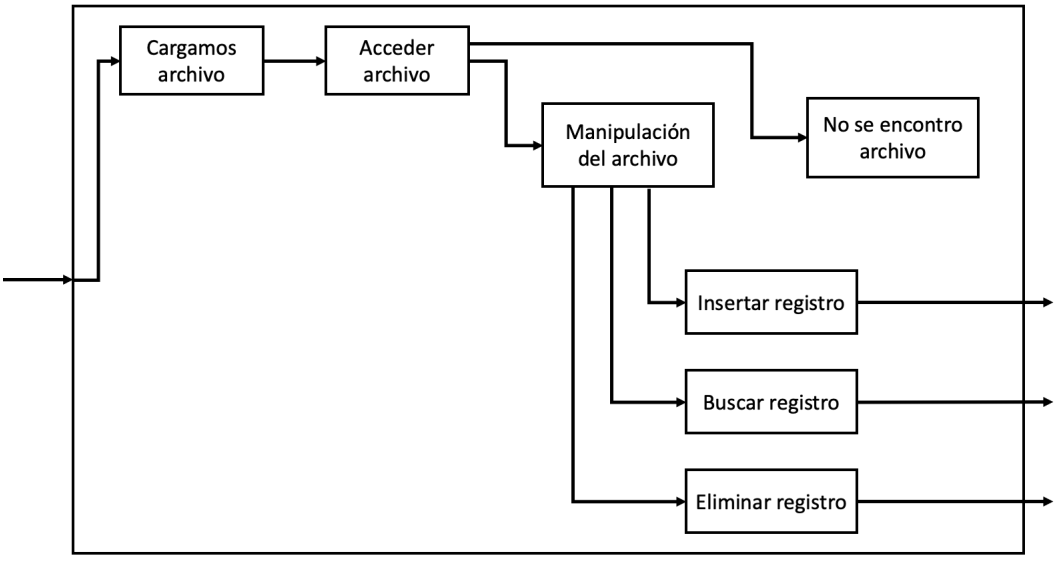
```
char id[5];
      char nombre[51];   //max 2147483647  6-digits
      char billions[7];    //3
      char country[31]; //4
      char industry[31]; //2
      char state[5]; //0: no eliminado, -1: fin de la cadena; >0: ↩
         pos logica del nextDel (Eliminados)
```

```
7       char next[5]; //Lista de ordenamiento
```

Listing 1: Gramática

2.2.1. Manipulación

Este módulo contiene una serie de funciones que permite la manipulación de una archivo ".txt". Para ello , describiremos el funcionamiento de esta máquina.



El procedimiento se explica como se muestra en el gráfico. Primero se carga el archivo al cual accederemos(nosotros para dar la primera prueba tomamos unos 30 registros), para posteriormente acceder al archivo deseado.Segundo, si logramos acceder al archivo, procedemos a ingresar a cualquiera de las funciones para poder alterar o mostrar el archivo.

Sequential File

Todo es un heap, desde la posicion 1 a n mantenemos los registros, de modo que matengan su posicion fisica, pero, cuando se añadimos registros n+1,n+2,....,n+k, se ordenan a tra vez del puntero, más no fisicamente.

- **Busqueda:** Nuestra función **"search"** se encarga de la búsqueda, tenemos que recorrer los registros almacenados con **"searchmain" y "searchaux**, si encontramos el registro solicitado lo retornara, caso contrario, comparamos lo que retornara las funciones ya mencionadas, de modo que nos quedamos el más cercano al registro solicitado, damos prioridad al mayor más cercano. Antes de empezar a analizar a mayor profundidad, tomemos en cuenta que "0" significa que "existe" y "-1" que "no existe".
  - **El "searchmain" utilizará el "binarySearch"**. El cual tiene diferentes casos.

- En el primer caso si existe y no está eliminado, te devolverá el registro que buscamos.

- El segundo caso, si existe, pero está eliminado, aquí hay dos posibilidades, la primera posibilidad te brindara el mayor más cercano, la segunda posibilidad es que no encuentre el mayor más cercano ya que están eliminados, así hasta llegar al inicio del archivo, motivo por el que buscará el menor más cercano.

- El tercer caso, no existe y no es el primero, sería similar al segundo caso, nos daría el mayor o menor más cercano.

- En el cuarto caso, si es mayor al primero, nos devolvería el menor más cercano.

- El **"searchaux"** haría una búsqueda secuencial, de modo que buscaría el registro que solicitamos, de caso no encontrarlo, buscaríamos el mayor o menor más cercano, de manera similar al **"binarySearch"**.
- También tenemos al **"searchBetween"** el cual nos muestra todos los registros que se encuentran entre dos parámetros, se ubican los registros con los punteros, los cuales no indican el orden.

- **Inserción:** Aquí encontramos a la función **"adding"**, la cual, primero verifica si el registro a ingresar existe, caso que existe no se añadirá nada, pero si no se encuentra, procederemos a añadirla, esto se añadirá dentro de nuestro temporal aux; nosotros delimitamos el tamaño máximo del aux antes de que se desborde; no se estará ordenando físicamente, pero nuestros punteros si se ordenarán.
  - El ordenamiento físico ocurre gracias a la función **"sortd"**, el cual estará verificando que no se desborde nuestro temporal aux, cuando se desborde, la función ordenara todo fisicamente, pero omitió los que ya este eliminados.
- **Eliminación:** La función **"eliminate"** es la encargada de eliminar los registros que se le solicite, al eliminar un registro, procedemos a reemplazar el orden de los punteros de modo que esté ordenado y listo para la siguientes operaciones

ISAM.-

Sus principales operaciones del ISAM denso:

- **Busqueda:** Tenemos un **"search"**, el cual accede a la función **"binarySearch"**, la cual se encarga de buscar de manera análoga al registro solicitado, si no es encontrado, devuelve el inmediato anterior. Una vez con el registro devuelto, el **"search"** se encarga de verificar si existe o no, ya que anteriormente no se verificó, si no existe buscará el mayor más cercano o en su defecto de no encontrarlo al menor más cercano.

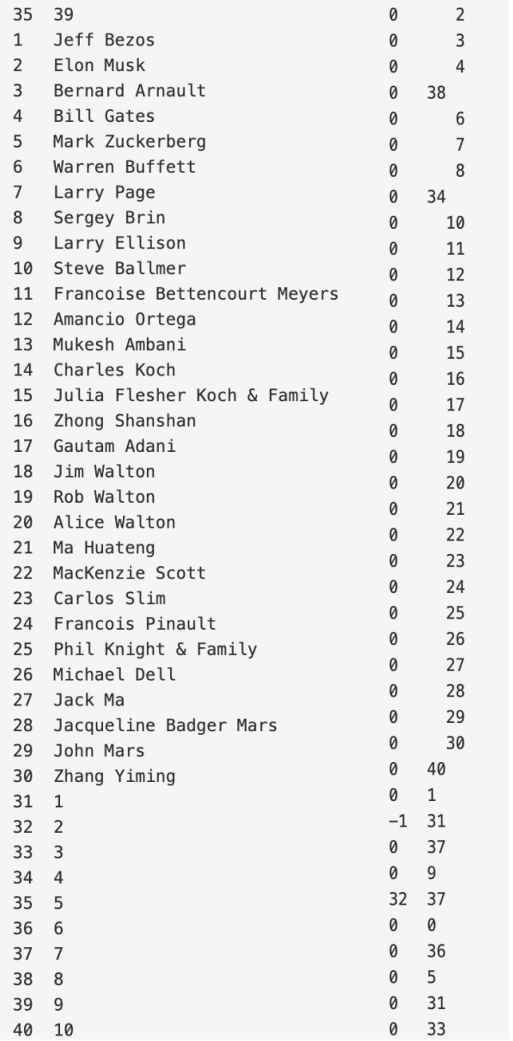
    - También tenemos al **"searchBetween"**, el cual es parecido al del sequential file, el cual devuelve los registros que se encuentren entre dos parámetros, pero, no devuelve registros que eliminados.

- **Inserción:** La función **"add"**, primero verifica si ya existe lo que se quiere añadir; no hay más de dos atributos con la misma llave, aquí siempre se buscará posicionar en el lugar más conveniente para ahorrar recursos, aprovechando los registros eliminados, pero para que se aproveche estos registros, se debe verificar el inmediato superior e inferior, si las condiciones cumple pasa a tomar su nuevo lugar, pero si no cumple y no se encuentran eliminados con los requerimientos, se enviara al final.
- **Eliminación:** Para la eliminación de registros tenemos a la función **"eliminate"**, la cual abre el index para verificar si existe o no el registro ingresado, si ya no existe, no pasa nada, pero si existe, tendremos que obtener el header, ir al registro que queremos eliminar, hacer una eliminación lifo, para poder escribir el nuevo header modificado.

3. Resultados

Sequential File:
Podremos observar en la cuarta columna a los punteros, los cuales apuntan a posiciones de los registros, estos punteros están designados de manera que los registros están ordenados, dependiendo del dinero que se muestra en cada registro. También podremos observar en la tercera columna si dichos registros están eliminados o no, los registros que posean un "0" significa que existen.

Los registros están divididos en la parte principal y el aux, la parte principal siempre se mantendra, solo sufrirá cambios sus punteros o la demostración de si existen o no, en cambio el aux inicialmente estará vacía, lugar donde se insertarán los nuevos registros, pero poseemos una función "sortd" la cual regula el aux, de manera que si se desborda, altera todo, de manera que ordena no sólo los punteros, también los altera físicamente, de manera que abría un nuevo orden, pero dejará de lado todo registro que ya no exista.



35	39	0	2
1	Jeff Bezos	0	3
2	Elon Musk	0	4
3	Bernard Arnault	0	38
4	Bill Gates	0	6
5	Mark Zuckerberg	0	7
6	Warren Buffett	0	8
7	Larry Page	0	34
8	Sergey Brin	0	10
9	Larry Ellison	0	11
10	Steve Ballmer	0	12
11	Francoise Bettencourt Meyers	0	13
12	Amancio Ortega	0	14
13	Mukesh Ambani	0	15
14	Charles Koch	0	16
15	Julia Flesher Koch & Family	0	17
16	Zhong Shanshan	0	18
17	Gautam Adani	0	19
18	Jim Walton	0	20
19	Rob Walton	0	21
20	Alice Walton	0	22
21	Ma Huateng	0	23
22	MacKenzie Scott	0	24
23	Carlos Slim	0	25
24	Francois Pinault	0	26
25	Phil Knight & Family	0	27
26	Michael Dell	0	28
27	Jack Ma	0	29
28	Jacqueline Badger Mars	0	30
29	John Mars	0	40
30	Zhang Yiming	0	1
31	1	-1	31
32	2	0	37
33	3	0	9
34	4	32	37
35	5	0	0
36	6	0	36
37	7	0	5
38	8	0	31
39	9	0	33
40	10		

4. Pruebas

El video donde probamos la organización de archivose encontrara dentro de un drive.