

ARCHIVOS RELATIVOS (DIRECTO)

Por: Univ. Nina Mamani Teodoro David

E-mail: dvd@pmail.umsanet.edu.bo

Resumen

En la organización de los archivos es muy necesario, pero una de las formas es del tipo archivos directos estos son accedidos directamente por medio de llaves que a través de un método son convertidas en direcciones en las que se almacenan los registros.

Ahora para que las conversiones de llave a direcciones se utilizan distintos métodos o técnicas. Pero una técnica es la función *HASH*, esta función consiste en tomar el residuo de la división de la llave y un número primo superior más cercano al número máximo de registro que almacena el archivo. No debemos de olvidar que esta función provoca colisiones, esto por la duplicidad de direcciones. Esto pasara con mayor frecuencia si el archivo esta lleno o cerca del numero máximo de registros, para esto se puede usar el factor de carga, esta indica que tan lleno esta nuestro archivo y que tan bueno es el desempeño. El calculo se lo realiza mediante la división de registros usados sobre numero máximo de registros que puede almacenar.

Tenemos que encontrar la forma de evitar colisiones para esto también existen métodos o técnicas, podríamos mencionar a encadenamiento de sinónimos,.... y existen muchas más.

Palabras Clave

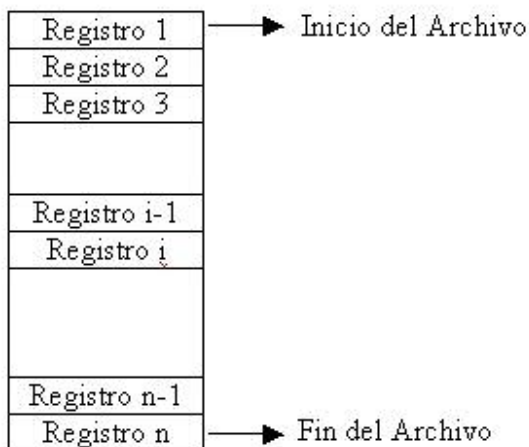
Estructura, Dato, Archivo, Registro, Directo, Inserción, Eliminación, Colisión.

Estructura de un archivo directo

Empecemos diciendo que una colección de registros de longitud fija será un archivo relativo (directo), y se almacena los registros uno al lado del otro en un dispositivo de almacenamiento de acceso directo. Este tipo de almacenamiento en estos archivos se lo realiza en discos, tomemos en cuenta que el acceso a los registros se hace generalmente en orden aleatorio.

Cada registro en un archivo de organización relativa se puede referir por medio de un número -entero- de dirección, el cual indica su distancia o desplazamiento desde el origen del archivo. Al primer registro en un archivo relativo se le asigna el valor 1, 2 al siguiente y así sucesivamente. De este modo, la dirección relativa de un valor entero que refleja su posición respecto al primer registro del archivo. El acceso aleatorio de un registro en un archivo de organización relativa se hace vía su número relativo de registro.

Un archivo de organización relativa puede crearse con un programa en un lenguaje de alto nivel si es que el método de acceso del sistema operativo central es capaz de manejar esta organización, y si el compilador del lenguaje de interfase con tal método de acceso.



Inserción, Eliminación y manejo de colisiones

Para poder insertar elementos en archivos relativos debemos de utilizar una técnicas que nos ayudaran en el calculo de direcciones, pero la mayoría de estas técnicas provocan las llamadas *colisiones*, son nada mas que repetición de direcciones. Pero estas colisiones no pueden ser eliminadas, pero podemos tratar de minimizarlas.

Pero antes un poco de explicación sobre este fenómeno, si las claves primarias de los registros son números consecutivos, se puede hacer un simple enlace directo entre claves y direcciones. Sencillamente se asocia una llave primaria más pequeña con la dirección relativa 1, la siguiente clave primaria más pequeña con la dirección relativa 2, y así sucesivamente.

Clave primaria	Dirección relativa
200	1
201	2
202	3

Asignación Simple

$$\text{Dirección relativa} = \text{clave primaria} - \text{clave más pequeña} + 1$$

Calculo de la dirección relativa

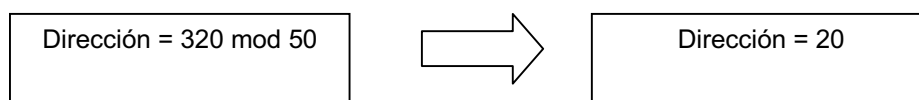
El ejemplo anterior no es muy bueno cuando los valores LLAVE no son consecutivos, o cuando el rango de las claves primarias es mucho más grande que el número de registros en el archivo.

La mejor solución para el problema anterior consistiría en utilizar una tabla de consulta, donde cada dirección relativa se asociaría a la clave primaria que le corresponde, por ejemplo:

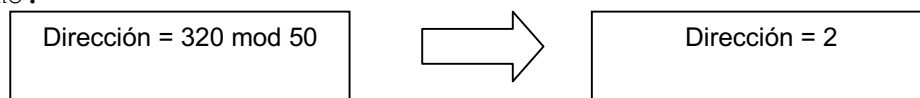
Clave primaria	Dirección relativa
2	3
2	8
666	2

Con la tabla de consulta podremos buscar la dirección relativa de acuerdo a la clave primaria que sea proporcionada. El problema con este tipo de método es el espacio que se requiere para almacenar la tabla de consulta especialmente si el archivo es de tamaño considerable.

Otra solución consiste en el cálculo de la dirección por medio de formulas matemáticas, un ejemplo de este tipo es el método de **HASHING POR RESIDUO** el cual consiste en dividir la llave por un *numero determinado* para obtener la dirección. Para determinar el número divisor se escoge el *numero máximo de registro* que almacenara el archivo, de esta forma el residuo siempre estará en el rango de registros que puede aceptar el archivo. Para ejemplificar lo anterior supongamos que tenemos un archivo con un máximo de registros de 50 y al introducir una llave igual a 320 entonces decimos que:



Entonces el registro será almacenado o localizado en la dirección 20. Pero como se hablo antes, esta clase de métodos producen colisiones, que este caso, van en aumento cuando el archivo comienza a saturarse. Existe una forma de aminorar la ocurrencia de colisiones en este método y consiste en tomar como numero divisor al numero primo mayor más cercano al numero máximo de registros que puede almacenar el archivo, en el caso anterior el numero divisor seria el 53 debido a que es mayor a 50 y es numero primo:



Otro punto que debemos considerar es el *factor de carga* que nos sirve para medir el grado de saturación y por lo tanto el grado de eficiencia que puede tener un archivo y se calcula de la forma siguiente:

$$\text{Factor de carga} = \frac{\text{Número de registros en el archivo}}{\text{Número máximo de registro en el archivo}}$$

Generalmente se dice que el factor de carga no debe ser mayor a 0.8 para considerar que el archivo aun tiene un buen nivel de desempeño.

Debido a que las colisiones no pueden ser erradicadas existen distintos métodos que funcionan para minimizar sus efectos, por ejemplo:

Sondeo Lineal

Cuando la posición del registro ya esta ocupada lo mas sencillo es hacer una búsqueda secuencial partiendo de la dirección original hasta encontrar una localidad vacía o libre. El PSEUDOCÓDIGO de este método se muestra a continuación:

```
INICIO
read( llave );
dir ← hash( llave );
DirOrigen ← dir;
HACER
{
  SI Dir ← Ocupada
  {
    dir ← dir+1
    SI dir > MaxRegistros
    {
      dir ← 1;
    }
  }
  SI NO
  {
    AlmacenarRegistro(dir);
    Salir;
  }
} MIENTRAS dir <> DirOrigen
print( "El archivo esta lleno " );
FIN.
```

Doble hash

Consiste en repetir la operación HASH para obtener un nuevo resultado. El doble hashing hace una dispersión de sinónimos a diferencia del sondeo lineal que por su naturaleza tiende a agrupar los sinónimos. El doble hash tiene un mejor desempeño para factores de carga menores a 0.5 y actúa mejor que el sondeo lineal con factores de carga mayores para búsquedas exitosas pero no así en búsquedas no exitosas.

Si deseas una aplicación puedes visitar "descargas" de :
URL: <http://www.geocities.com/exesoftbolivia>

Referencia bibliográfica

- *Lenguaje C y Estructura de Datos*, García de Solaé J. & Garcerán Vicente [D32629]©
- *Programación Estructurada en C*, James L Antonakos[D32651]©
- *Estructuras de Datos : Prog. y Apli.*, Lewisé T.G. & Smithé M.Z.[E1 002]©
- *Algoritmos + Estructuras de Datos=Programas*, Wirthé N.[E1 005]©
- *Algoritmos y Estructura de Datos*, Wirthé Niklaus[E1 029]©
- *Estructura de Datos*, Choque A.& Casilla G. C[E1 008]©
- *Estructuras de Archivos*, Folké Michael & Zoellické Bill[E1 019]©
- *Estructura de la Información Organización de Ficheros*, Muñoz López Francisco J.[E1 024]©

[DVD] 01/09/03