Tema 7: Organización Encadenada

Contenidos extraídos del libro:

Ficheros. Organizaciones clásicas para el almacenamiento del a información

Autores: *Irene Luque Ruiz, Juan Antonio Romero del Castillo y Miguel Ángel Gómez-Nieto*
Editorial: Servicio de Publicaciones de la Universidad de Córdoba, 1998.
ISBN 84-7801-468-3

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.1 Introducción

El fichero encadenado:

- Los ficheros representan objetos y clases de objetos del mundo real
- Dichos objetos están relacionados entre sí en el mundo real
- Dichas relaciones deben ser representadas también en los ficheros
- Atributos especialmente indicados para representar relaciones entre registros de distintos ficheros: **PUNTEROS**

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.1 Introducción

PUNTEROS:

- Definición: un atributo cuya misión es “apuntar” a otro registro del mismo o de distinto fichero.
- “apuntar”: hacer referencia a un registro de forma física o lógica.
- Tipos de punteros:
  1. Físicos: valor correspondiente a cilindro, pista y sector donde se encuentra el registro.
  2. Relativos: la posición relativa del registro con respecto al comienzo del fichero.
  3. Simbólicos: valor simbólico mediante el cual se puede obtener alguno de los anteriores.

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.1 Introducción

PUNTEROS:

Físicos

- Son más rápidos para el acceso al registro al que apuntan.
- Inconvenientes:
  - Dependen del dispositivo de almacenamiento.
  - Ocupan mucho pues tienen mucha información que almacenar.
  - Dependen de la posición del registro en el fichero.

Relativos

- Independientes del dispositivo físico.
- Inconvenientes:
  - Siguen dependiendo de la posición del registro en el fichero.

Simbólicos

- Independientes del dispositivo físico y posición del registro.
- Más intuitivos.
- Inconvenientes:
  - Más complicados de manejar pues se requiere su conversión a uno de los anteriores.

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.1 Introducción

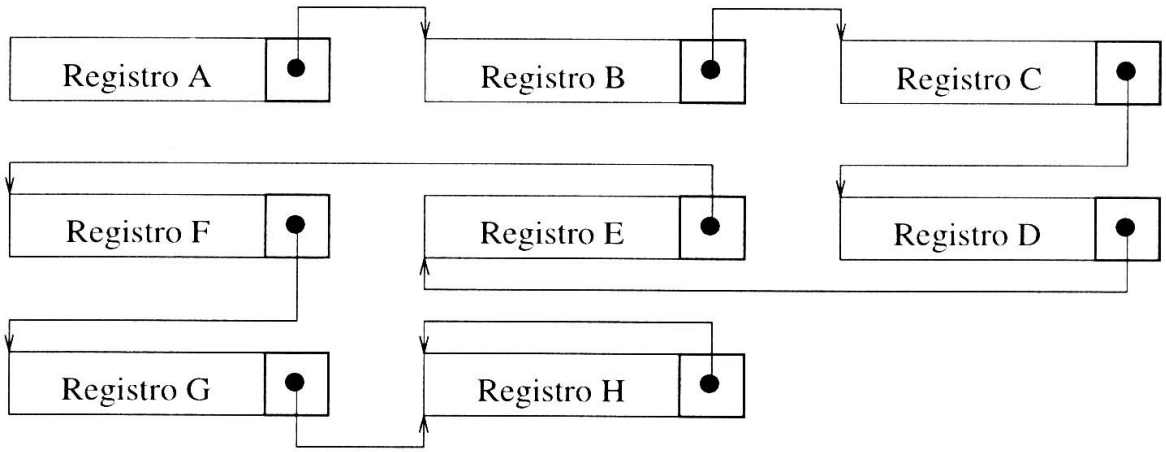
- **Definición de fichero encadenado:** aquel en el que uno de sus atributos es un puntero.
- **Punteros Externos:** relacionan registros de distintos ficheros.
- **Punteros Internos:**
  - Un registro hace referencia a otro dentro del mismo fichero.
  - Pueden establecer uno o varios órdenes internos de los registros.
- **Zona Maestra:** los registros pueden organizarse de diversas formas: apilo, secuencial, etc.

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.2 Estructura de las cadenas

La ZONA MAESTRA:

- Los registros encadenados pueden tener longitud fija o variable.
- concepto de **cadena**: serie de registros enlazados mediante punteros.
- Existe la *cabeza* y la *cola* de la cadena.
- Su orden en la cadena lo impone el predicado de ordenación seleccionado para esa cadena.
- Parecido a las clásicas estructuras dinámicas: *listas*



7.2 Estructura de las cadenas

Ejemplo: Los pacientes diarios

En el sistema del Laboratorio de Análisis Clínicos, se desea mantener información sobre los pacientes que acceden diariamente al Laboratorio. La primera vez que un paciente acude al Laboratorio se le abre una ficha con sus datos personales –aquellos que se consideran que no son dependientes con los análisis que se van a realizar– y cada vez que acude se le da de alta en el archivo denominado ***`diario.dat`*** *en el que se desea almacenar: la identificación del paciente, el doctor que prescribe el conjunto de análisis y la compañía médica a la cual se le facturará el coste del análisis.*

7.2 Estructura de las cadenas

Ejemplo: Los pacientes diarios

Tenemos **DOS** posibles soluciones:

1.- En cada registro de **`diario.dat`** se almacena información acerca de la compañía médica:

a. Redundancia de la información.

b. Posibles incoherencias si cambian los datos de la compañía pues hay que modificarlos en todos los registros donde aparezca.

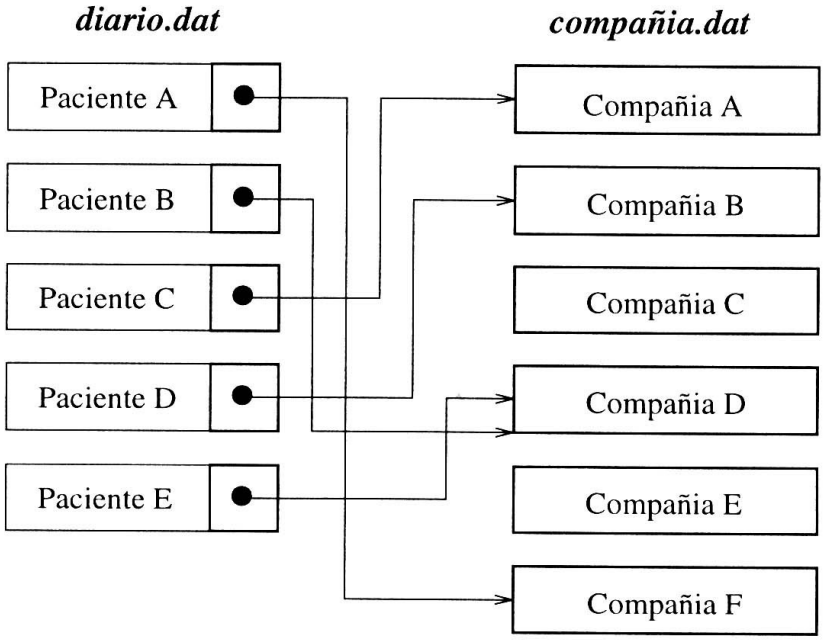
2.- En cada registro de **`diario.dat`** se almacena un puntero a un registro de la compañía médica en un fichero **`compañía.dat`**

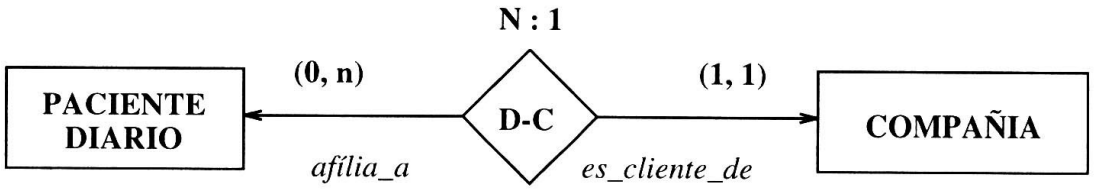
a. No hay redundancia de la información.

b. No hay posibilidad de incoherencias.

7.2 Estructura de las cadenas

Ejemplo: Los pacientes diarios





© Depto. Informática y Análisis Numérico. Universidad de Córdoba

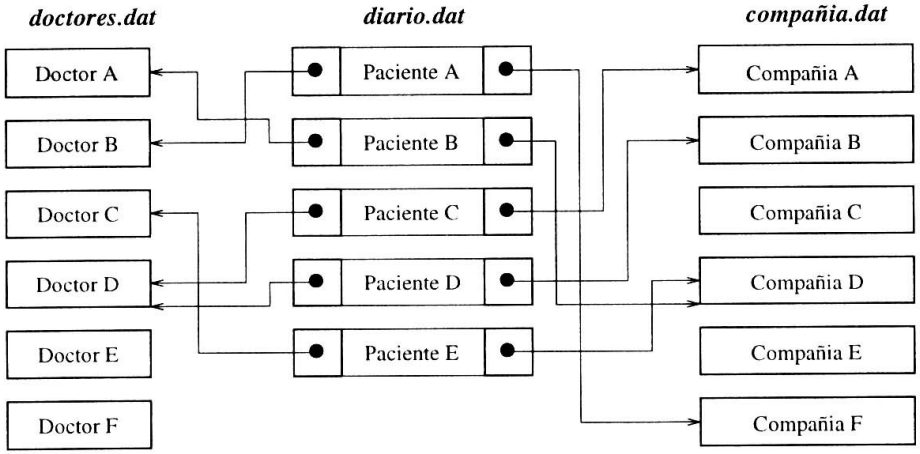
7.2 Estructura de las cadenas

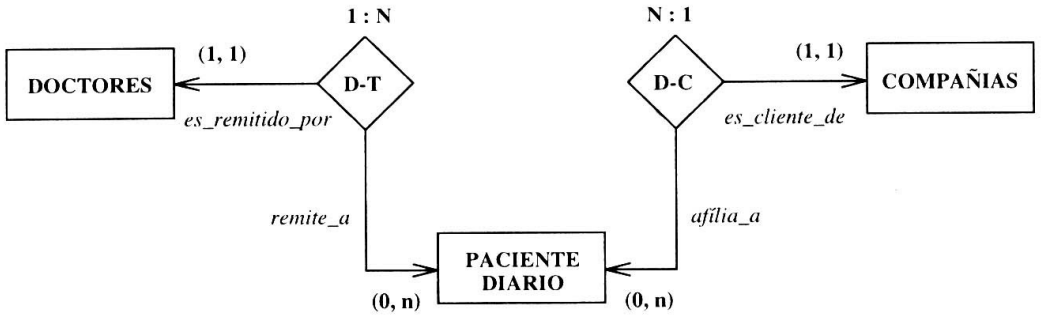
Ejemplo: Los doctores

Al laboratorio le interesa mantener información sobre todos los doctores que le envían pacientes para la realización de pruebas, principalmente con dos fines: (a) mantener "mailing" de doctores para enviarles información acerca del Laboratorio, sus técnicas de análisis, servicios que ofrece, etc. Y, (b) para, si el doctor lo desea, enviarle por correo los resultados de los análisis de los pacientes que el doctor ha prescrito. La información correspondiente a los doctores (nombre y apellidos, dirección, teléfono, número de colegiado, etc.) se encuentra almacenada en un archivo denominado **`doctores.dat`**, *de cuya estructura no nos preocuparemos por el momento.*

7.2 Estructura de las cadenas

Ejemplo: Los doctores





De nuevo:

- No hay redundancia de la información
- No hay posibilidad de incoherencias

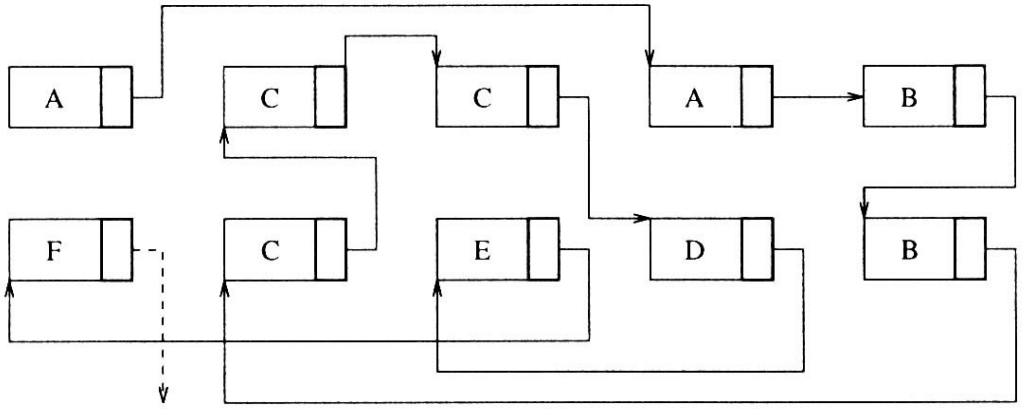
7.2 Estructura de las cadenas

Ejemplo: Las pruebas diarias

A cada paciente que acude al Laboratorio se le va a realizar una serie de análisis según la prescripción del doctor. Esta información debe almacenarse de tal forma que el acceso a la misma en base a los análisis que se realizan en el Laboratorio sea eficiente. Se decide, por tanto, almacenar la información correspondiente a los análisis que realiza el Laboratorio cada día en un archivo encadenado, denominado ***`resultados.dat`****, en el que los registros se encuentren ordenados en base al nombre o identificación del análisis. Cada registro debe almacenar información correspondiente al parámetro analítico, paciente al cual se le realiza y resultado de la prueba.*

7.2 Estructura de las cadenas

Ejemplo: Las pruebas diarias



una cadena

Similitudes y diferencias con la organización secuencial:

= Los registros están ordenados por una clave

≠ Pero el orden físico no se corresponde con el lógico

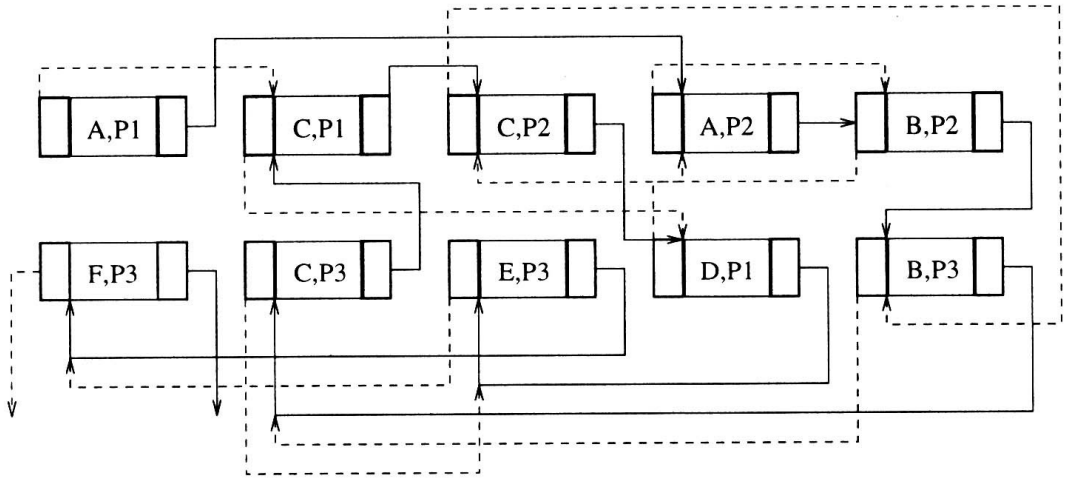
7.2 Estructura de las cadenas

Ejemplo: Las pruebas diarias

- Los accesos por identificador de prueba son eficientes, pero ¿Qué ocurre si queremos acceder a todos los análisis de un paciente?

  → Sería necesaria la lectura de toda la cadena

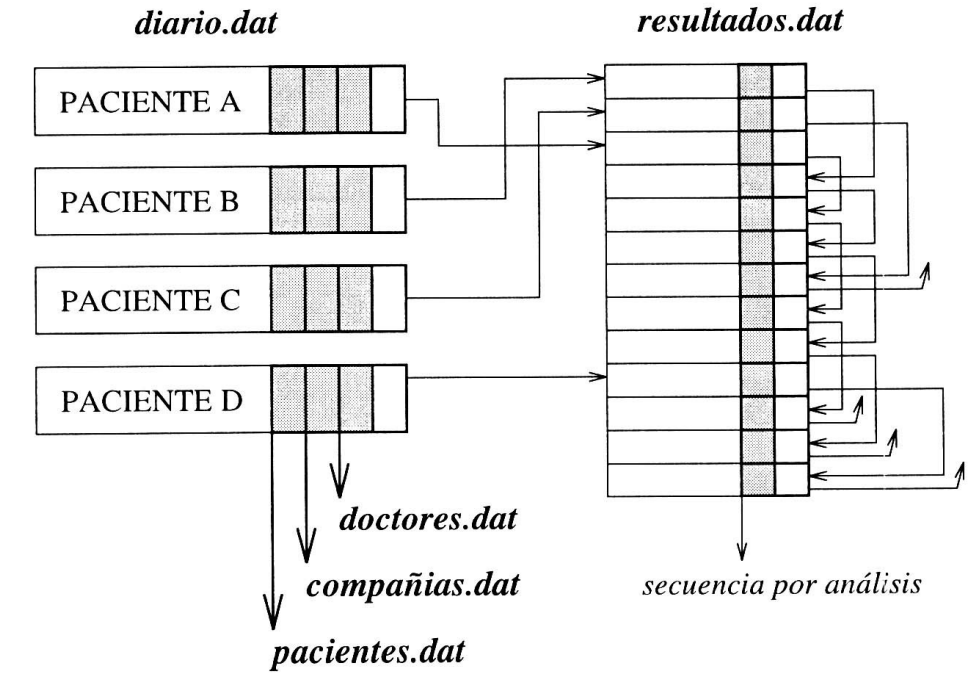
- Solución: Añadir una segunda cadena:

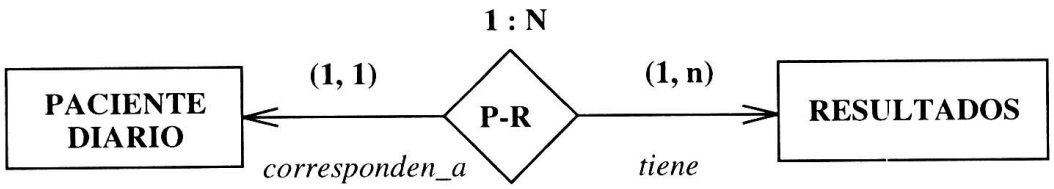


dos cadenas

7.2 Estructura de las cadenas

Ejemplo: Las pruebas diarias

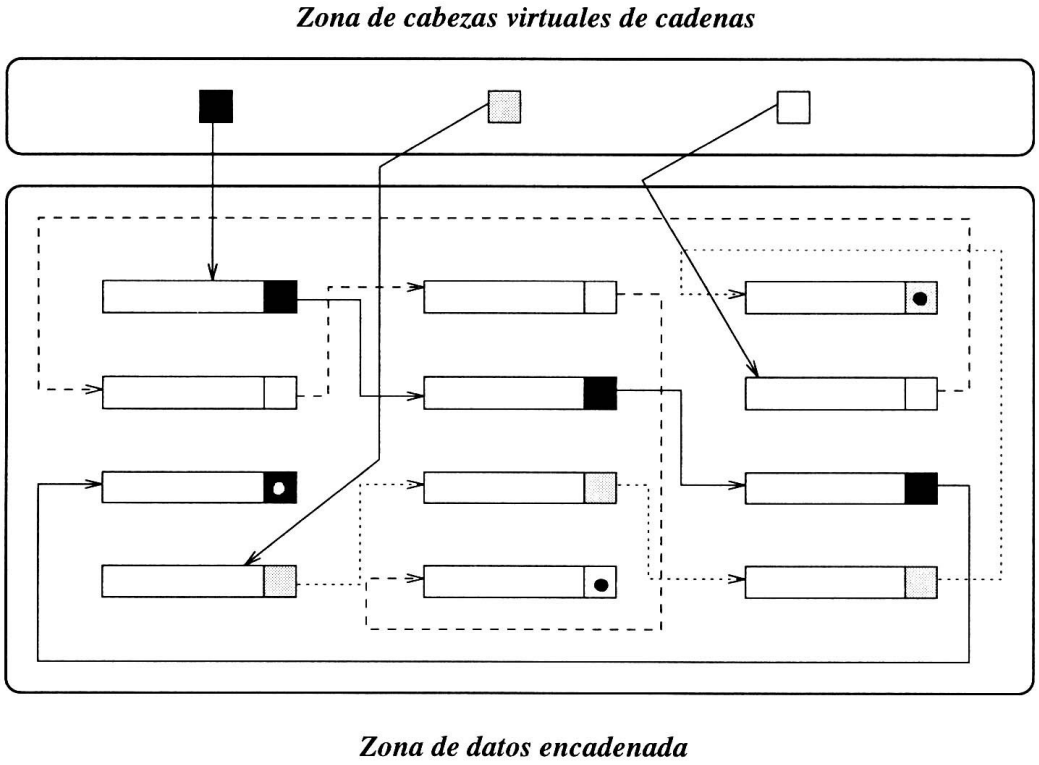




7.2 Estructura de las cadenas

Es necesario mantener el primer elemento de cada cadena:

- Registro con la cabeza de cada cadena
- En el mismo o en distinto archivo



7.3 Ocupación

Fichero Encadenado (igual apilo estructurado)

longitud variable, tamaño medio de registro

$$\bar{R} = \sum_{j=1}^{k} \left(\bar{L_{V_j}} + L_S \right)$$

longitud variable, ocupación del fichero

$$\Gamma = \sum_{i=1}^{r} \sum_{j=1}^{k} \left(L_{V_{ij}} + L_{S_j} \right)$$

longitud fija, tamaño de registro

$$R = \sum_{j=1}^{k} L_{V_j}$$

incluyendo el tamaño de los punteros

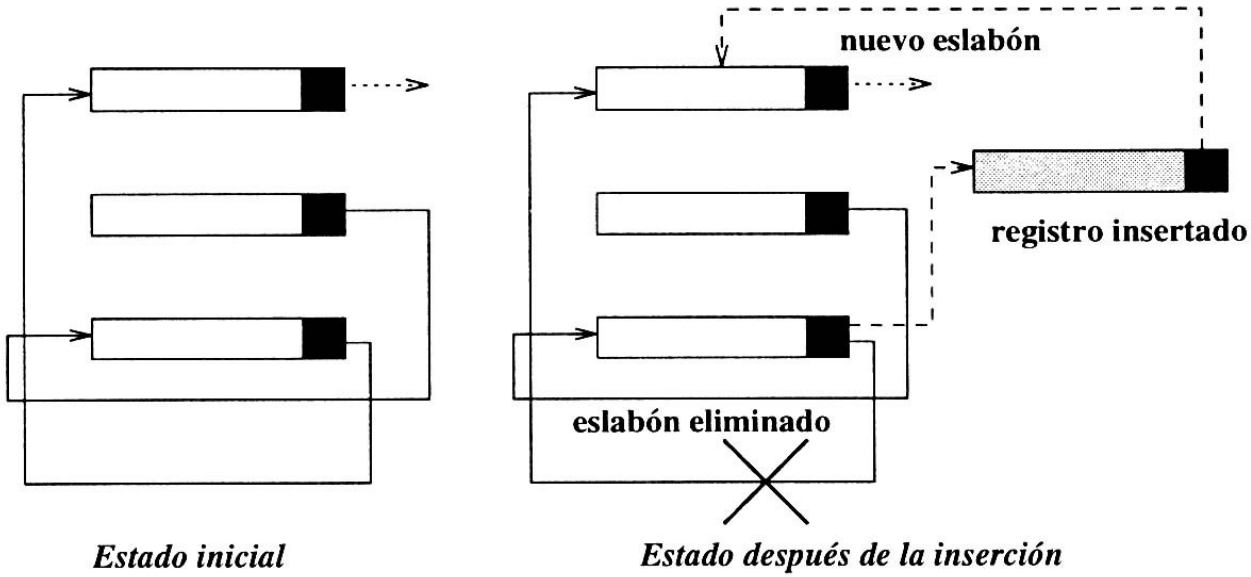
longitud fija, ocupación del fichero

$$\Gamma = R \times r$$

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.4 Operaciones con ficheros encadenados

Inserción: T_I



7.4 Operaciones con ficheros encadenados

Inserción: T_I

1. Localizar posición de inserción del nuevo registro.
2. Acceder al registro “precedente” y guardar el puntero que almacena.
3. Modificar el puntero del registro “precedente” para que apunte al nuevo registro insertado.
4. Modificar el registro nuevo con el valor del puntero que tenía el registro “precedente”.
5. Escritura del nuevo registro en el archivo.

$$T_I = T_{I_0} + c \times (T_L + T_{RE} + T_P)$$

T_{I_0}: tiempo de inserción del nuevo registro (depende de la org.)

c: (*minúscula*) número de cadenas en las que interviene el nuevo registro.

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.4 Operaciones con ficheros encadenados

Lectura: T_L

- Siempre en base a un criterio de búsqueda
- Si el criterio no es en base a la clave, la lectura tarda igual que en la organización apilo:

$$T_L = t_l + \frac{١}{٢}(C+١)(t_r + T_{tB}) \approx \frac{C}{٢}(t_r + T_{tB})$$

C: (*mayúscula*) tamaño de la cadena. No confundir con '*c*' minúscula

7.4 Operaciones con ficheros encadenados

Lectura: T_L

- Si los registros de la cadena se encuentran en bloques adyacentes:

$$T_L = \frac{1}{2}\frac{C}{F_B}\left(t_r + T_{tB}\right)$$

7.4 Operaciones con ficheros encadenados

Lectura Consecutiva: T_{LC}

- Como el registro actual tiene el puntero al siguiente:

$$T_{LC} = t_r + T_{tB}$$

- Si los registros están agrupados en bloques adyacentes solo cuando leemos el último, el siguiente está en otro bloque:

$$\overline{T}_{LC} = \frac{1}{F_B}\left(t_r + T_{tB}\right)$$

7.4 Operaciones con ficheros encadenados

Actualización: T_A

- Si no supone la modificación del atributo por el cual está ordenada una de las cadenas

$$T_A = T_L + T_{RE} + T_P$$

7.4 Operaciones con ficheros encadenados

Actualización: T_A

Si supone la modificación del atributo por el cual está ordenada una de las cadenas:

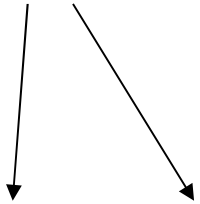
1. Lectura del registro objeto de la modificación y de su precedente para modificarlo y que señale al nuevo registro siguiente.

2. Encontrar el nuevo registro "precedente" y reasignar su siguiente y el del registro modificado.

7.4 Operaciones con ficheros encadenados

Actualización: T_A

Por cada cadena afectada hay que leer dos registros y re-escribirlos modificados



$$T_A = T_L + T_{I_0} + c' \times (2T_L + 2T_{RE} + T_P)$$

T_{I_0}: tiempo de inserción del nuevo registro, depende de la org.

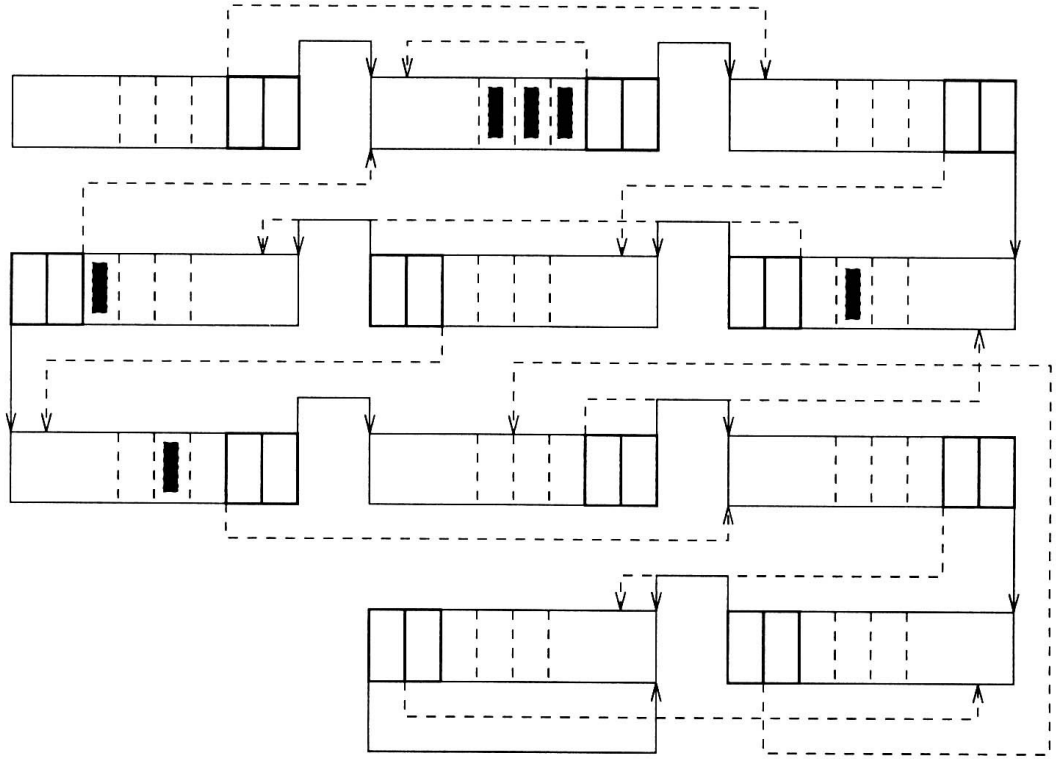
c': (*prima*) número de cadenas que se ven afectadas por la actualización del registro

7.4 Operaciones con ficheros encadenados

Actualización: T_A

La actualización es costosa:

- Es mejor marcar para borrado
- Habrá tantas marcas de borrado como cadenas



© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.4 Operaciones con ficheros encadenados

Lectura Exhaustiva (total): T_{LT}

Supone leer todos los datos sin importar el orden:

- Todos los registros del fichero (igual que apilo):

$$T_{LT} = bT_{tB} = rT_{tR}$$

- Todos los registros de una cadena (hay que recorrerla entera):

$$T_{LT} = T_{L_0} + (C-1)T_{LC}$$

T_{L_0}: tiempo de acceso a la cabeza de la cadena

Puede ser muy costoso

7.4 Operaciones con ficheros encadenados

Lectura Ordenada T_{LO}

Supone leer todos los datos en un orden:

- Si el orden lo pone una cadena, es igual que leer exhaustivamente dicha cadena:

$$T_{LO} = T_{LT}$$

- Si el orden es por otro atributo diferente al de las cadenas, hay que ordenar el fichero

$$T_{LO} = T_C(C) + T_{LT}$$

! Muy Costoso !

7.4 Operaciones con ficheros encadenados

Reorganización: T_{RO}

- Las operaciones de actualización y borrado generan registros marcados para borrado
- Volatilidad de los datos: cuando los datos almacenados dejan de ser útiles debido a una operación (borrado, actualización, etc.)
- Pérdida o corrupción de los punteros debido a algún problema del sistema
- Su complejidad depende del número de cadenas existente: más cadenas, mayor complejidad

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.4 Operaciones con ficheros encadenados

Reorganización de una sola cadena: T_{RO}

El proceso incluye:

1. Leer todos los registros
2. Despreciar los marcados para borrado
3. Ordenar los registros modificando el atributo puntero de cada registro para que apunte al siguiente registro en el orden
4. Renombrar el nuevo fichero

$$T_{RO} = T_{LT} + T_C(r) + b'(t_r + T_{tB} + T_{RE})$$

r: número de registros del fichero

b': número de bloques del nuevo fichero

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.4 Operaciones con ficheros encadenados

Reorganización si existe más de una cadena: T_{RO}

El proceso incluye:

1. Reorganizar el archivo como se ha descrito anteriormente por la cadena principal
2. Para cada cadena se construye un archivo auxiliar con todos los registros de esa cadena ordenados
3. Se busca en el fichero inicial cada registro y se establecen los valores adecuados a los punteros

$$T_{RO_c} \approx T_C(r) + T'_{LT}(C) + C(T_L + T_{RE} + T_P)$$

T'_{LT} : lectura exhaustiva del fichero clasificado

7.4 Operaciones con ficheros encadenados

Reorganización total (suma): T_{RO}

$$T_{RO} = T_{RO_0} + \sum_{i=2}^{c} T_{RO_{Ci}}$$

T_{RO_0}: tiempo de reorganización de la cadena principal

$T_{RO_{Ci}}$: tiempo de reorganización de la cadena secundaria i-ésima

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.5 Estructura de las cadenas multidirección

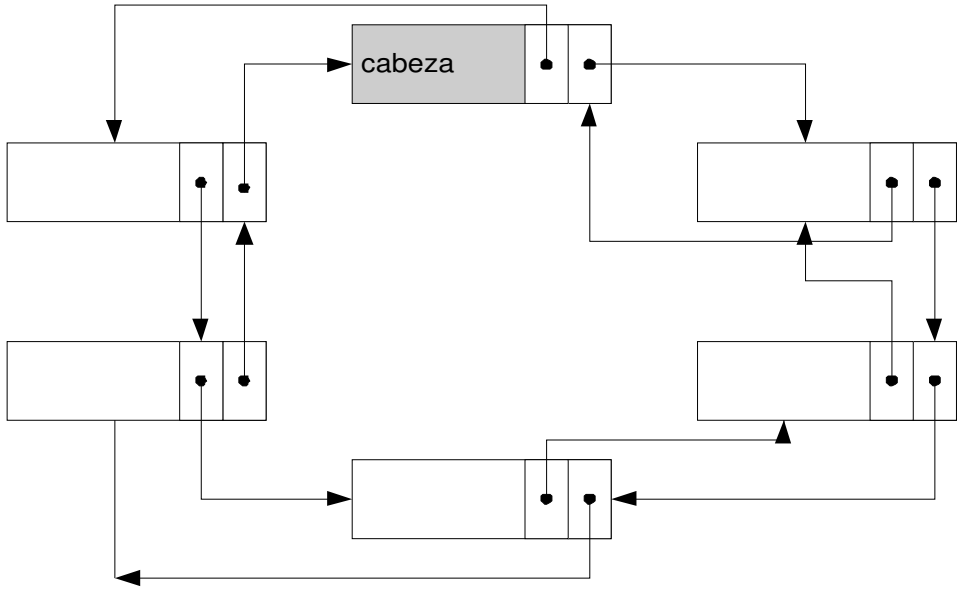
- Una cadena es tan débil como su eslabón más débil y esto supone un gran riesgo
- En las operaciones con punteros es frecuente la corrupción de los punteros, pérdida, etc., y esto afectará a todo el fichero
- La solución son las **cadenas multidireccionales**
- Cada registro tiene varios punteros a registros de la misma cadena además de al “siguiente”
- Otro cometido de estas cadenas es mejorar el acceso con algoritmos para acceder de distintas formas

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

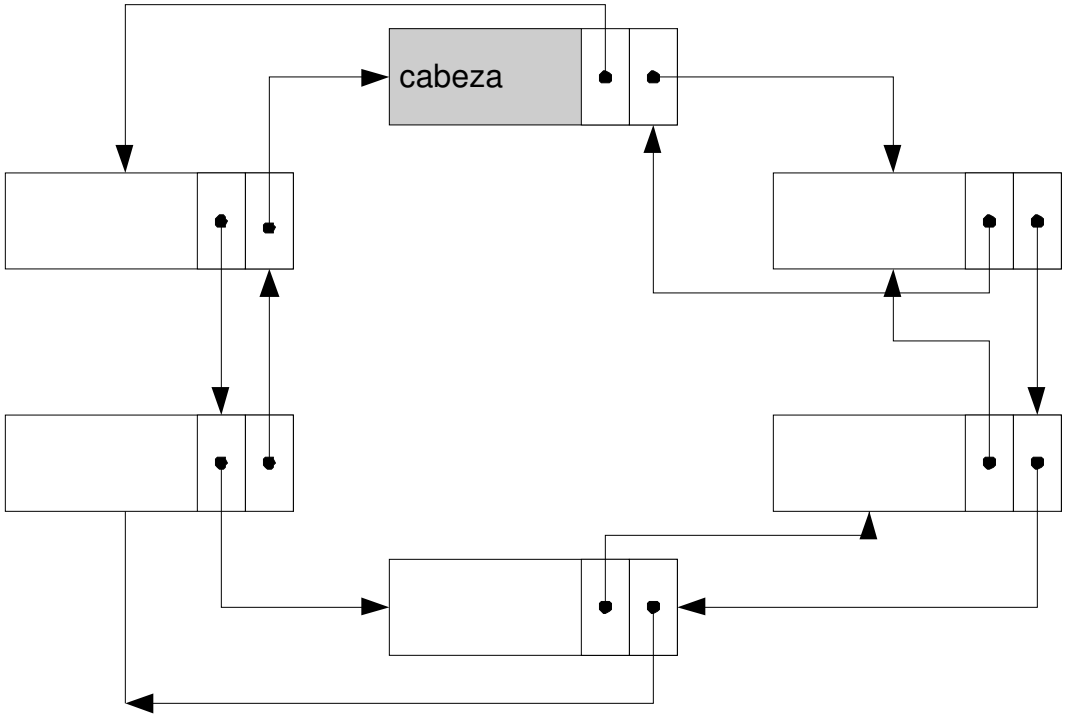
7.5 Estructura de las cadenas multidirección

Cadenas bidireccionales:

- Hay un puntero al registro “siguiente” y otro al registro “precedente”
- Existe una “cabeza” y una “cola” de la cadena
- Organización en **Anillo**:
  - El “precedente” de la “cabeza” es el último registro
  - El “siguiente” de la “cola” es el primer registro



7.5 Estructura de las cadenas multidirección



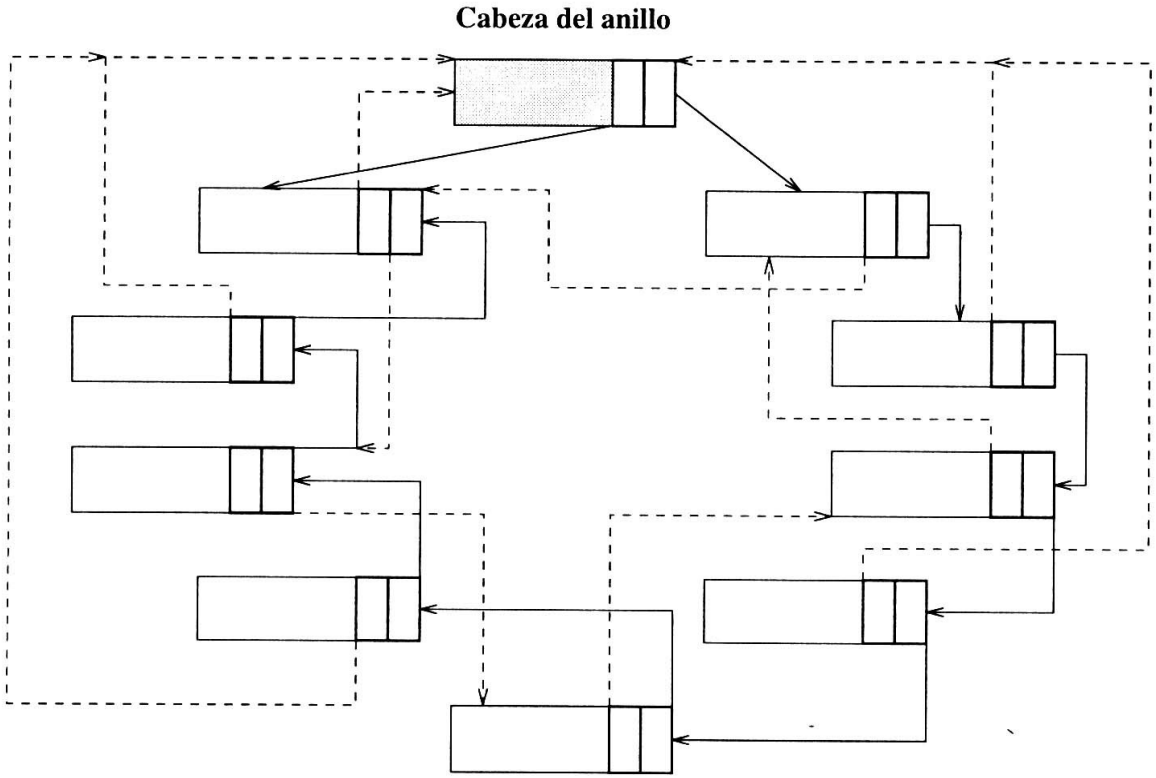
Ventajas del Anillo:

- Aunque algún puntero se deteriore, puede accederse a todos los registros con el otro puntero
- Existen dos direcciones y el algoritmo de recorrido puede elegir distintas formas de hacerlo

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.5 Estructura de las cadenas multidirección

Estructura en **Anillo de coral**



- Ahora un puntero hace referencia al “siguiente” y otro al anterior al “precedente” o a la “cabeza” de la cadena.
- Son igual de seguros que los anteriores y mejoran ciertos tipos de recorridos pues permiten un avance mayor (de más de un registro) en ciertas direcciones y acceder siempre a la cabeza
- Existen otras muchas variantes

7.5 Estructura de las cadenas multidirección

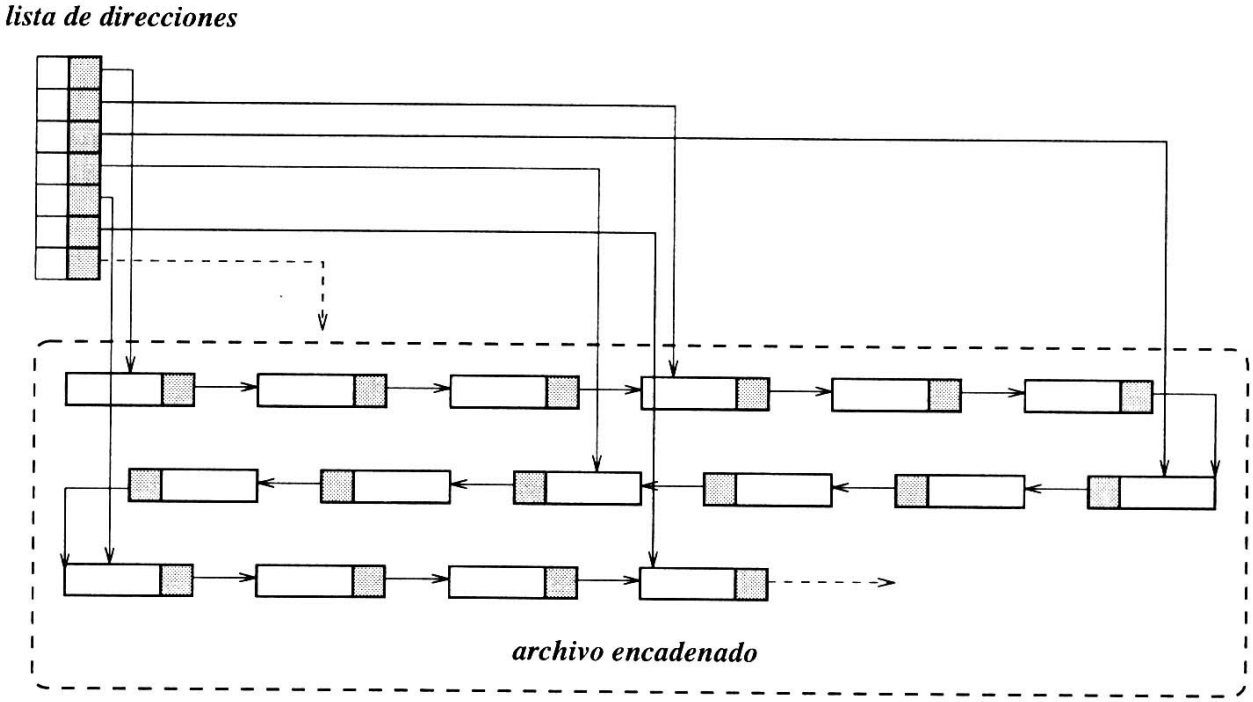
Segmentación de cadenas:

- En una cadena segmentada cada registro tiene un puntero al “siguiente” (también puede tener al “precedente”)
- Y además tiene un puntero “lejano” a registros alejados en más de una posición formando así: **segmentos**
- Se puede avanzar rápidamente en los recorridos con estos punteros “lejanos” y cuando se llega cerca del registro buscado recorrer con los punteros al “siguiente”

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.5 Estructura de las cadenas multidirección

Lista de direcciones:



- En lugar de que los punteros estén empotrados en los registros, se habilita una tabla que almacena pares: atributo-puntero
- La zona maestra puede ser apilada y la tabla es la que permite el acceso eficiente
- Si la lista es pequeña puede alojarse en memoria permanentemente
- Los atributos de la lista pueden estar ordenados mejorándose así su propio acceso

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

7.6 Resumen

- Los ficheros encadenados permiten representar relaciones complejas entre tipos de entidades del mundo real mediante **punteros**.
- Muy usadas como base de otras pues son potentes y aminoran el coste de mantener información relacionada
- Permiten definir varios órdenes internos en un mismo fichero
- Pueden presentar problemas de pérdida de información si las cadenas son simples y los punteros se deterioran

© Depto. Informática y Análisis Numérico. Universidad de Córdoba

Fin