Clase 2

Archivos - Algorítmica Clásica

- Agregar nuevos elementos
- Actualizar un archivo maestro con uno o varios archivos detalles
- Corte de control
- Merge de varios archivos

Archivos. Ej 4 Agregar Datos a un archivo existente

■ Precondiciones
– Se procesa un solo archivo
– Ya tiene información
– Se le incorporan datos nuevos
– El proceso muestra como se hace
– Al ser un procedimiento las declaraciones necesarias están en el programa principal

```
Procedure agregar (Var Emp: Empleados);
  var E: registro;

  begin
    reset( Emp );
    seek( Emp, filesize(Emp));
    leer( E );
    while E.nombre <> ' ' do begin
        write( Emp, E );
        leer( E );
    end;
    close( Emp );
  end;
```

Archivos ➔ Actualización Maestro Detalle

Este problema involucra utilizar al mismo tiempo varios archivos de datos.

• Se denomina maestro al archivo que resume un determinado conjunto de datos.
• Se denomina detalle al que agrupa información que se utilizará para modificar el contenido del archivo maestro.
• En general
    • Un maestro
    • N detalles.

Consideraciones del proceso (precondiciones)

• Ambos archivos (maestro y detalle) están ordenados por el mismo criterio
• En el archivo detalle solo aparecen empleados que existen en el archivo maestro
• Cada empleado del archivo maestro a lo sumo puede aparecer una vez en el archivo detalle

Archivos ➔ Actualizar Un Maestro con Un detalle

```
program actualizar;
  type emp = record
          nombre: string[30];
          direccion: string[30];
          cht: integer;
        end;
        e_diario = record
            nombre: string[30];
            cht: integer;
        end;
  detalle = file of e_diario;
  maestro = file of emp;
 var regm: emp;
     regd: e_diario;
     mae1: maestro;
     det1: detalle;
```

```
begin
    assign (mae1, 'maestro');
    assign (det1, 'detalle');

    reset (mae1);
    reset (det1);

    while (not eof(det1)) do begin
        read(mae1, regm);
        read(det1,regd);

        while (regm.nombre <> regd.nombre) do
          read (mae1,regm);

        regm.cht := regm.cht + regd.cht;
        seek (mae1, filepos(mae1)-1);
        write(mae1,regm);
      end;
  end.
```

Precondiciones del ejemplo

•Ambos archivos (maestro y detalle) están ordenados por código del producto)
•En el archivo detalle solo aparecen productos que existen en el archivo maestro
•Cada producto del maestro puede ser, a lo largo del día, vendido más de una vez, por lo tanto, en el archivo detalle pueden existir varios registros correspondientes al mismo producto

Archivos ➔ Un Maestro Un detalle sobre la base del anterior

```
program actualizar;
  const valoralto='9999';
  type str4 = string[4];
       prod = record
         cod: str4;
         descripcion: string[30];
         pu: real;
         cant: integer;
       end;
       v_prod = record
         cod: str4;
         cv: integer;
       end;
       detalle = file of v_prod;
       maestro = file of prod;
  var
    regm: prod;
    regd: v_prod;
    mae1: maestro;
    det1: detalle;
    total: integer;
```

```
begin
    assign (mae1, 'maestro');
    assign (det1, 'detalle');

    reset (mae1);  reset (det1);

    while (not eof(det1)) do begin
        read(mae1, regm);
        read(det1,regd);
        while (regm.cod <> regd.cod) do
          read (mae1,regm);
        while (regm.cod = regd.cod) do begin
          regm.cant := regm.cant - regd.cv;
          read (det1,reg);
        end;
        seek (mae1, filepos(mae1)-1);
        write(mae1,regm);
      end;
  end.
```

Solucion correcta.

```
procedure leer (var archivo:detalle;
                var dato:v_prod);
    begin
      if (not eof(archivo))
        then read (archivo,dato)
        else dato.cod := valoralto;
    end;
```

```
begin
    assign (mae1, 'maestro');
    assign (det1, 'detalle');
    reset (mae1);  reset (det1);
    leer(det1,regd);
    while (regd.cod <> valoralto) do begin
        read(mae1, regm);
        while (regm.cod <> regd.cod) do
          read (mae1,regm);
        while (regm.cod = regd.cod) do begin
          regm.cant := regm.cant - regd.cv;
          leer(det1,regd);
        end;
        seek (mae1, filepos(mae1)-1);
        write(mae1,regm);
    end;
End;
```

Archivos ➔ Un Maestro N detalle

- El problema siguiente generaliza aún más el problema anterior
- El maestro se actualiza con tres archivos detalles
- Los archivos detalle están ordenados de menor a mayor
- Condiciones de archivos iguales, misma declaración de tipos del problema anterior

```
procedure leer (var archivo: detalle;
                var dato:v_prod);
 begin
   if (not eof(archivo))
     then read (archivo,dato)
     else dato.cod := valoralto;
 end;
```

```
Var   regm: prod;  min, regd1, regd2,regd3: v_prod;
      mae1: maestro;   det1,det2,det3: detalle;
begin
  assign (mae1, 'maestro');   assign (det1, 'detalle1');
  assign (det2, 'detalle2');  assign (det3, 'detalle3');
  reset (mae1);  reset (det1); reset (det2); reset (det3);

  leer(det1, regd1); leer(det2, regd2); leer(det3, regd3);

  minimo(regd1, regd2, regd3, min);
  while (min.cod <> valoralto) do  begin
      read(mae1,regm);
      while (regm.cod <> min.cod) do read(mae1,regm);
      while (regm.cod = min.cod ) do begin
         regm.cant:=regm.cant - min.cantvendida;
         minimo(regd1, regd2, regd3, min);
      end;
      seek (mae1, filepos(mae1)-1);
      write(mae1,regm);
  end;
end.
```

```
procedure minimo (var r1,r2,r3: v_prod;
                  var min:v_prod);
 begin
   if (r1.cod<=r2.cod) and
      (r1.cod<=r3.cod) then begin
      min := r1;  leer(det1,r1)
    end
    else if (r2.cod<=r3.cod) then begin
           min := r2; leer(det2,r2)
         end
         else begin
           min := r3;leer(det3,r3)
         end;
 end;
```

Archivos ➔ Corte de control

El problema consiste en lageneración de reportes

•Es un problema clásico en el manejo de BD.

•Si bien los DBMS lo manejan diferente, veremos la algorítmica clásica de los mismos

•Precondiciones

•El archivo se encuentra ordenado por provincia, partido y ciudad

Provincia: xxxx
Partido: yyyy
Ciudad # Var. # Muj. Desocupados
aaa ....
bbb ....
ccc ....
Total Partido .....
Partido: zzzz
Ciudad # Var. # Muj. Desocupados
... .
Total Partido .....
Total Provincia: .....
Provincia: qqqq

```
program Corte_de_Control;
  const valoralto='zzzz';
  type str10 = string[10];
       prov = record
         provincia, partido, ciudad: str10;
         cant_varones,
         cant_mujeres,
         cant_desocupados : integer;
       end;
       instituto = file of prov;
  var regm: prov;
      inst: instituto;
      t_varones, t_mujeres,
      t_desocupa, t_prov_var,
      t_prov_muj, t_prov_des: integer;
      ant_prov, ant_partido : str10;
```

```
procedure leer (var
   archivo:instituto;  var dato:prov)
  begin
   if (not eof( archivo ))
    then read (archivo,dato)
    else dato.provincia := valoralto;
  end;
```

```
begin
    assign (inst, 'censo' );
    reset (inst);

    leer (inst, regm);

    writeln ('Provincia: ',regm.provincia);
    writeln ('Partido: ', regm.partido);
    writeln('Ciudad','Mas','Fem','Desocupa');

    t_varones := 0;
    t_mujeres := 0;
    t_desocupa := 0;
    t_prov_var := 0;
    t_prov_muj := 0;
    t_prov_des := 0;
```

```
ant_partido := regm.partido;
if (ant_prov <> regm.provincia) then
   begin
    writeln ('TotalProv.',t_prov_var,
       t_prov_muj, t_prov_des);
    t_prov_var := 0; t_prov_muj := 0;
    t_prov_des := 0;
    writeln('Prov.:',regm.provincia);
   end;
   writeln('Partido:', regm.partido);
 end;
end.
```

```
while ( reqm.provincia <> valoralto)do begin
   ant_prov := reqm.provincia;
   ant_partido := reqm.partido;
   while (ant_prov=reqm.provincia) and
         (ant_partido=reqm.partido) do begin
      write (reqm.ciudad, reqm.cant_varones,
        reqm.cant_mujeres,reqm.cant_desocupados);
      t_varones := t_varones + reqm.cant_varones;
      t_mujeres := t_mujeres + reqm.cant_mujeres;
      t_desocupa := t_desocupa + reqm.cant_desocupados;
      leer (inst, reqm);
   end;
   writeln ('Total Partido: ', t_varones, t_mujeres,
   t_desocupa);
   t_prov_var := t_prov_var + t_varones;
   t_prov_muj := t_prov_muj +  t_mujeres;
   t_prov_des := t_prov_des + t_desocupa;
   t_varones := 0; t_mujeres := 0; t_desocupa := 0;
```

Archivos ➔ Merge

Involucra archivos con contenido similar, el cual debe resumirse en un único archivo.

Precondiciones:

- Todos los archivos detalle tienen igual estructura
- Todos están ordenados por igual criterio

Primer ejemplo:

- Conceptos de Programación inscribe a los alumnos que cursarán la materia en tres computadoras separadas. C/U de ellas genera un archivo con los datos personales de los estudiantes, luego son ordenados físicamente por otro proceso. El problema que tienen los JTP es genera un archivo maestro de la asignatura
  - **Precondiciones**
    - El proceso recibe tres archivos con igual estructura
    - Los archivos están ordenados por nombre de alumno
    - Un alumno solo aparece una vez en el archivo
  - **Postcondición**
    - Se genera el archivo maestro de la asignatura ordenado por nombre del alumno

```
program union_de_archivos;
  const valoralto = 'zzzz';
  type str30 = string[30];
       str10 = string[10];
       alumno = record
         nombre: str30;
         dni: str10;
         direccion: str30;
         carrera: str10;
       end;
       detalle = file of alumno;
  var min,regd1,regd2,regd3: alumno;
      det1,det2,det3,maestro : detalle;
procedure leer (var archivo:detalle; var dato:alumno);
    begin
      if (not eof( archivo ))
        then read (archivo, dato)
        else dato.nombre := valoralto;
    end;
```

```
procedure minimo (var r1,r2,r3:alumno; var
                  min:alumno);
   begin
     if (r1.nombre<r2.nombre) and
        (r1.nombre<r3.nombre) then begin
            min := r1;
            leer(det1,r1)
       end
     else if (r2.nombre<r3.nombre) then begin
               min := r2;
               leer(det2,r2)
             end
          else begin
               min := r3;
               leer(det3,r3)
          end;
   end;
```

```
begin
    assign (det1, 'det1'); assign (det2, 'det2'); assign (det3, 'det3');
    assign (maestro, 'maestro');
    rewrite (maestro);
    reset (det1);    reset (det2);    reset (det3);
    leer(det1, regd1); leer(det2, regd2)  leer(det3, regd3);
    minimo(regd1, regd2, regd3, min);
    while (min.nombre <> valoralto) do
      begin
        write (maestro,min);
        minimo(regd1,regd2,regd3,min);
      end;
    close (maestro);
end.
```

■ Los vendedores de cierto comercio asientan las ventas realizadas …..
■ Precondiciones
– Similar al anterior
– Cada vendedor puede realizar varias ventas diarias

```
program union_de_archivos_II;
  const valoralto = '9999';
  type str4 = string[4];
       str10 = string[10];
       vendedor = record
         cod: str4;
         producto: str10;
         montoVenta: real;
       end;
       ventas = record
         cod: str4;
         total: real;
       end;
       detalle = file of vendedor;
       maestro = file of ventas;
```

```
var min, regd1, regd2, regd3: vendedor;
      det1, det2, det3: detalle;
      mae1: maestro;
      regm: ventas;
      aux: str4;
```

```
begin
    assign (det1, 'det1'); assign (det2, 'det2'); assign (det3, 'det3');
    assign (mael, 'maestro');

    reset (det1);    reset (det2);     reset (det3);
    rewrite (mael);

    leer (det1, regd1);     leer (det2, regd2);     leer (det3, regd3);
    minimo (regd1, regd2, regd3,min);

    while (min.cod <> valoralto) do begin
        regm.cod := min.cod;
        regm.total := 0;
        while (regm.cod = min.cod ) do begin
            regm.total := regm.total- min.montoVenta;
            minimo (regd1, regd2, regd3, min);
        end;
        write(mael, regm);
     end;
End;
```

Archivos ➔ Merge N archivos con repetición

Los vendedores de cierto comercio asientan las ventas realizadas…

Precondiciones

- Similar al anterior
- Cada vendedor puede realizar varias ventas diarias

Idem anterior con N archivos….

Archivos ➔ Merge N archivos con repetición

```
program union_de_archivos_III;
  const valoralto = '9999';
  type vendedor = record
         cod: string[4];
         producto: string[10];
         montoVenta: real;
       end;
       ventas = record
         cod: string[4];
         total: real;
       end;
     maestro = file of ventas;
     arc_detalle=array[1..10] of file of vendedor;
     reg_detalle=array[1..10] of vendedor;
  var min: vendedor;
      deta: arc_detalle;
      reg_det: reg_detalle;
      mael: maestro;
      regm: ventas;
      i,n: integer;
```

```
procedure leer (var archivo:detalle; var
dato:vendedor);
    begin
      if (not eof( archivo ))
        then read (archivo, dato)
        else dato.cod := valoralto;
    end;

procedure minimo (var reg_det: reg_detalle;
var min:vendedor; var deta:arc_detalle);
    var i: integer;
    begin
      { busco el minimo elemento del
        vector reg_det en el campo cod,
        supongamos que es el índice i }
      min = reg_det[i];
      leer( deta[i], reg_det[i];
    end;
```

```
begin
  Read(n)
  for i:= 1 to n do begin
    assign (deta[i], 'det'+i);
    { ojo lo anterior es incompatible en tipos}
    reset( deta[i] );
    leer( deta[i], reg_det[i] );
  end;
  assign (mael, 'maestro'); rewrite (mael);
  minimo (reg_det, min, deta);
  while (min.cod <> valoralto) do begin
        regm.cod := min.cod;
        regm.total := 0;
        while (regm.cod = min.cod ) do begin
          regm.total := regm.total+
                        min.montoVenta;
          minimo (regd1, regd2, regd3, min);
        end;
        write(mael, regm);
      end;
  end.
```

Clase 3

Archivos ➔ Introducción

La memoria primaria (RAM) es rápida y de simple acceso, pero su uso tiene algunas desventajas respecto al almacenamiento secundario:

- Capacidad limitada
- Mayor costo
- Es volátil

Archivos ➔ Introducción

Almacenamiento secundario necesita más tiempo para tener acceso a los datos que en RAM

- Su acceso es tan "lento" que es imprescindible enviar y recuperar datos con inteligencia
- Al buscar un dato, se espera encontrarlo en el primer intento (o en pocos)
- Si se buscan varios datos, se espera obtenerlos todos de una sola vez

La información está organizada en archivos

- Archivo: colección de bytes que representa información

Archivos ➔ Viaje de un Byte

Archivo Físico

- Archivo que existe en almacenamiento secundario
- Es el archivo tal como lo conoce el S.O. y que aparece en su directorio de archivos

Archivo Lógico

- Es el archivo, visto por el programa
- Permite a un programa describir las operaciones a efectuarse en un archivo,
- No se sabe cual archivo físico real se utiliza o donde esta ubicado

Archivos – Viaje de un byte

- Viaje de un byte ➔No es sencillo
  - Escribir un dato en un archivo
    - Write (archivo, variable) ➔ ciclos para escribir
- Quienes estan involucrados
  - Administrador de archivos
  - Buffer de E/S
  - Procesador de E/S
  - Controlador de disco

Archivos – Viaje de un byte

Administrador de archivos: conjunto de programas del S.O. (capas de procedimientos) que tratan aspectos relacionados con archivos y dispositivos de E/S

•En Capas Superiores: aspectos lógicos de datos (tabla)

- •Establecer si las características del archivo son compatibles con la operación deseada (1)

•En Capas Inferiores: aspectos físicos (FAT)

- •Determinar donde se guarda el dato (cilíndro, superficie, sector) (2)
- •Si el sector está ubicado en RAM se utiliza, caso contrario debe traerse previamente. (3)

Archivos – Viaje de un byte

Buffers de E/S: agilizan la E/S de datos.

• Manejar buffers implica trabajar con grandes grupos de datos en RAM , para reducir el acceso a almacenamiento secundario

Procesador de E/S: dispositivo utilizado para la transmisión desde o hacia almacenamiento externo. Independiente de la CPU. (3)

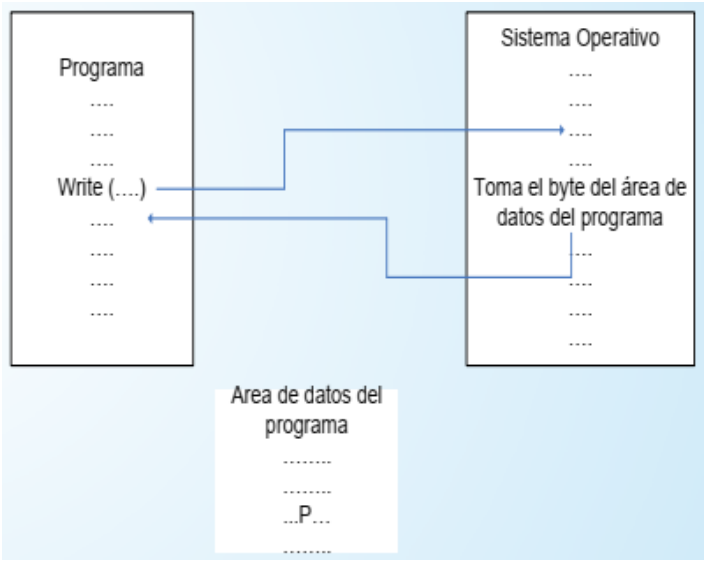
Archivos – Viaje de un byte

Controlador de disco: encargado de controlar la operación de disco.

• Colocarse en la pista
• Colocarse en el sector
• Transferencia a disco

Archivos – Viaje de un byte

- Qué sucede cuando un programa escribe un byte en disco?
  - Operación
  - Write(……)
  - Veamos los elementos que se involucran en esta simple operación
  - Supongamos que se desea agregar un byte que representa el carácter ‘P’ almacenado en una variable c de tipo carácter, en un archivo denominado TEXTO que se encuentra en algún lugar del disco rígido.

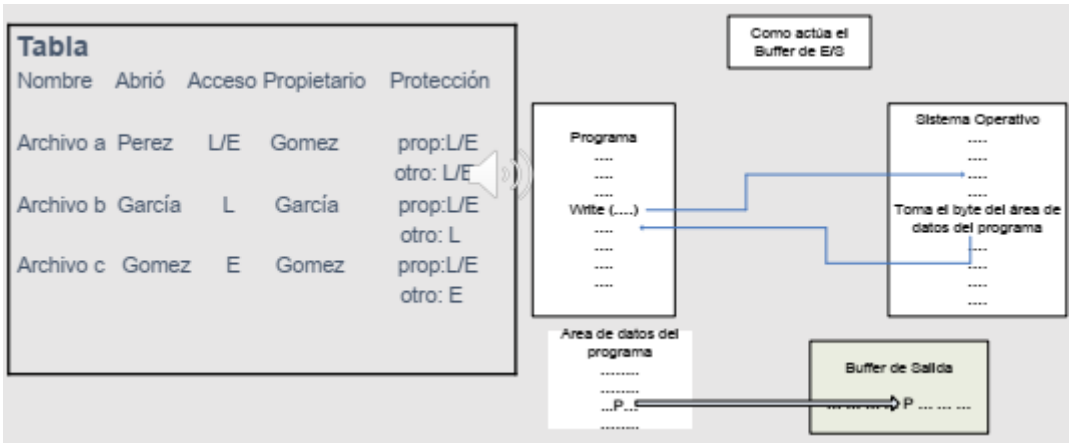


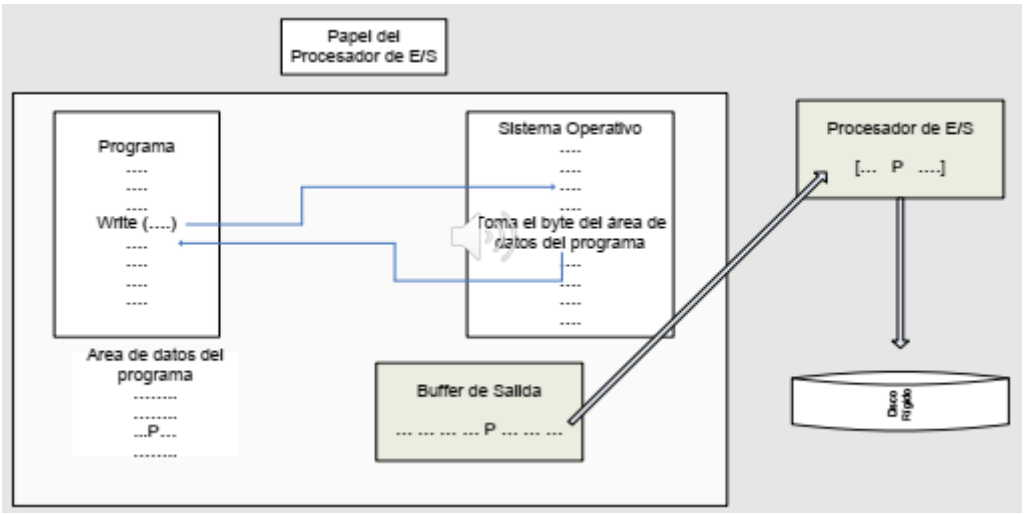
Archivos – Viaje de un byte

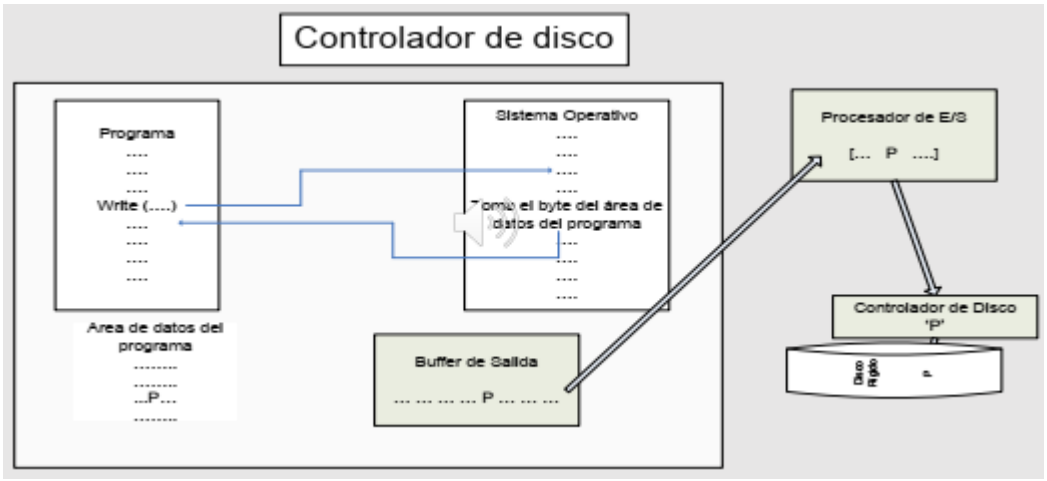
Capas del protocolo de transmisión de un byte

•El Programa pide al S.O. escribir el contenido de una variable en un archivo
•El S.O. transfiere el trabajo al Administrador de archivos
•El Adm. busca el archivo en su tabla de archivos y verifica las características
•El Adm. obtiene de la FAT la ubicación física del sector del archivo donde se guardará el byte.
•El Adm se asegura que el sector del archivo está en un buffer y graba el dato donde va dentro del sector en el buffer

•El Adm. de archivos da instruccciones al procesador de E/S (donde está el byte en RAM y en que parte del disco deberá almacenarse)
•El procesador de E/S encuentra el momento para transmitir el dato a disco, la CPU se libera
•El procesador de E/S envía el dato al controlador de disco (con la dirección de escritura)
•El controlador prepara la escritura y transfiere el dato bit por bit en la superficie del disco.







Archivos ➔ Tipos de Archivo

Archivos como Secuencia de bytes

• No se puede determinar fácilmente comienzo y fin de cada dato.
• Ejemplo: archivos de texto

Archivos estructurados

• Registros

    - Longitud fija o variable
- Campos
    - Longitud fija o variable

Archivos ➔ Tipos de Archivo

Campos

- Unidad lógicamente significativa más pequeña de un archivo. Permite separar la información
- Pueden ser:
    - Longitud fija
    - Cuales son?
- Longitud variable
    - Como determinar su longitud?

Archivos ➔ Tipos de Archivo

Campos de longitud variable

- Identidad de campos: variantes, pro y contras.
    - Longitud predecible (long. Fija), desperdicio de espacio, si el tamaño es pequeño al agrandarlo se podría desperdiciar más espacio)
    - Indicador de longitud (al ppio de cada campo)
    - Delimitador al final de cada campo (carácter especial no usado como dato)

Archivos ➔ Tipos de Archivo

Registros

- Organización de registros
- Longitud predecible (en cant. de bytes o cant. de campos)
    - Campos fijos o variables
- Longitud variable
    - Indicador de longitud (al comienzo, indica la cant. de bytes que contiene)
    - Segundo archivo (mantiene la info de la dirección del byte de inicio de cada registro)
    - Delimitador (carácter especial no usado como dato)
- Long. Predecible de registros
- Estudio de casos: ventajas y desventajas

Archivos ➔ Claves

- Se concibe al Registro como la cantidad de info. que se lee o escribe
- Objetivo: extraer sólo un registro específico en vez del archivo completo
- Es conveniente identificar una registro con una llave o clave que se base en el contenido del mismo

Clave

- Permite la identificación del registro
- Deben permitir generar orden en el archivo por ese criterio

Únivoca / Primaria:

- Identifican un elemento particular dentro de un archivo

Secundaria

- Reconocen un conjunto de elementos con igual valor

Archivos ➔ Claves

Forma canónica: forma estándar para una llave, puede derivarse a partir de reglas bien definidas.

- Representación única para la llave, ajustada a la regla
    - Ej: llave sólo con letras mayúsculas y sin espacios al final.
- Al introducir un registro nuevo:
    - 1ro se forma una llave canónica para ese registro
    - 2do se la busca en el archivo. Si ya existe, y es univoca ➔ no se puede ingresar

Archivos ➔ Claves (performance)

Estudio de performance

•Punto de partida para futuras evaluaciones
•Costo: acceso a disco, Nº de comparaciones
•Caso promedio

En el caso secuencial

• Mejor caso: leer 1 reg. , peor caso leer n registros
• Promedio: n/2 comparaciones
• Es de O(n), porque depende de la cantidad de registros
• Lectura de Bloques de registros
• mejora el acceso a disco,
• no varían las comparaciones.

Acceso directo

• Permite acceder a un registro preciso
• Requiere una sola lectura para traer el dato [O(1)].
• Debe necesariamente conocerse el lugar donde comienza el registro requerido

Número relativo de registro (NRR):

• Indica la posición relativa con respecto al principio del archivo
• Solo aplicable con registros de longitud fija)
    • Ej. NRR 546 y longitud de cada registro 128 bytes ➔ distancia en bytes= 546 * 128 = 69.888

El acceso directo es preferible sólo cuando se necesitan pocos registros específicos, pero este método NO siempre es el más apropiado para la extracción de info.

• Ej. generar cheques de pago a partir de un archivo de registros de empleados.
    • Como todos los reg. se deben procesar ➔ es más rápido y sencillo leer registro a registro desde el ppio. hasta el final, y NO calcular la posición en cada caso para acceder directamente.

Archivos ➔ diferentes visiones

Forma de acceso-Cantidad de cambios

Archivos ➔ Tipos

Forma de acceso

• Serie cada registro es accesible solo luego de procesar su antecesor, simples de acceder
• Secuencial los registros son accesibles en orden de alguna clave
• Directo se accede al registro deseado

de Cambios

• Estáticos -> pocos cambios
    • Puede actualizarse en procesamiento por lotes
    • No necesita de estructuras adicionales para agilizar los cambios
• Volátiles -> sometido a operaciones frecuentes:
    • Agregar / Borrar / Actualizar
    • Su organización debe facilitar cambios rápidos
    • Necesita estructuras adicionales para mejorar los tiempos de acceso

Archivos ➔ Operaciones

Altas Bajas Modificaciones Consultas Como influye registros de long. Fija y variable

Archivos ➔ eliminación

Eliminar registros de un archivo

• Baja física
• Baja lógica

- Cuales son las diferencias?
- Cuales las ventajas y desventajas?

Eliminar

• Cualquier estrategia de eliminación de registros debe proveer alguna forma para reconocerlos una vez eliminados (ejemplo: colocar una marca especial en el reg. eliminado).
• Con este criterio se puede anular la eliminación facilmente.
• Cómo reutilizar el espacio de registros eliminados ?
• Los programas que usan archivos deben incluir cierta lógica para ignorar los registros eliminados

Compactación.

• Recuperar el espacio
• La forma más simple es copiar todo en un nuevo archivo a excepción de los registros eliminados ➔ Baja Física
• Frecuencia
  - Tiempo (depende del dominio de aplicación)
  - Ante la necesidad de espacio
• Veremos el análisis de recuperación dinámica del almacenamiento

Aprovechamiento de espacio

• Reg. Longitud fija ➔ es necesario garantizar:
• Marca especiales en los reg. Borrados➔Baja Lógica
• Registros de longitud variable ➔ los nuevos elementos deben "caber" en el lugar

Recuperación del espacio para su reutilización cuando se agreguen registros

•Búsqueda secuencial -> usa las marcas de borrado.
  - •Para agregar, se busca el 1º reg. eliminado. Si no existe se llega al final del archivo y se agrega allí.
  - •Es muy lento para operaciones frecuentes.
- •Es necesario
  - •Una forma de saber de inmediato si hay lugares vacíos en el archivo
  - •Una forma de saltar directamente a unos de esos lugares, en caso de existir

Aprovechamiento de espacio (reg. long. fija)

• Recuperación de espacio con Lista o pilas (header)
  - • Lista encadenada de reg. disponibles.
  - • Al insertar un reg. nuevo en un archivo de reg. con long. fija, cualquier registro disponible es bueno.
  - • La lista NO necesita tener un orden particular, ya que todos los reg. son de long. fija y todos los espacios libres son iguales

•Recuperación de espacio con Lista o pilas (header)
  - •Ej : en el encabezado estará NRR 4, el archivo tendrá
    - •alfa beta delta * 6 gamma * -1 epsilon
  - •Se borra beta, como inicial quedará 2
    - •alfa * 4 delta * 6 gamma * -1 epsilon
  - •Si se quiere agregar un elemento el programa solo debe chequear el header y desde ahí obtiene la dirección del primero. Agrego omega , como ppio queda 4 nuevamente
    - •alfa omega delta * 6 gamma * -1 epsilon

Archivos - Eliminación

Aprovechamiento de espacio

- Recuperación de espacio con reg. de longitud variable
  - Marca de borrado al igual que en reg. de long. Fija (ej:*)
  - El problema de los registros de longitud variable está en que no se puede colocar en cualquier lugar, para poder ponerlo debe caber, necesariamente.
  - Lista. No se puede usar NRR como enlace. Se utiliza un campo binario que explícitamente indica en enlace (conviene que indique el tamaño).
  - Cada registro indica en su inicio la cant. de bytes.
  - Reutilización: buscar el registro borrado de tamaño adecuado (lo suficientemente grande).
  - Como se necesita buscar, no se puede organizar la lista de disponibles como una pila.
  - El tamaño "adecuado" del primer registro borrado a reutilizar ➔origina Fragmentación

Aprovechamiento de espacio ➔Fragmentación

- Interna: ocurre cuando se desperdicia espacio en un registro, se le asigna el lugar pero no lo ocupa totalmente.
  - Ocurre, en general, con reg. long. Fija.
  - Reg.long. Variable evitan el problema
  - Solución ➔ el "residuo" una vez ocupado el espacio libre, pasa a ser un nuevo reg. Libre. Si éste es muy chico (no se podrá ocupar)➔fragmentación externa

- Externa: ocurre cuando el espacio que no se usa es demasiado pequeño como para ocuparse. Soluciones:
  - Unir espacios libres pequeños adyacentes para generar un espacio disponible mayor (unir los huecos en el espacio de almacenamiento)
  - Minimizar la fragmentación, eligiendo el espacio más adecuado en cada caso.
- Estrategias de colocación en registros de longitud variable:
  - Primer ajuste
  - Mejor ajuste
  - Peor ajuste

- Primer ajuste: se selecciona la primera entrada de la lista de disponibles, que pueda almacenar al registro, y se le asigna al mismo.
  - Minimiza la búsqueda
  - No se preocupa por la exactitud del ajuste
- Mejor ajuste: elige la entrada que más se aproxime al tamaño del registro y se le asigna completa.
  - Exige búsqueda
- Peor ajuste: selecciona la entrada más grande para el registro, y se le asigna solo el espacio necesario, el resto queda libre para otro registro

Conclusiones

- Las estrategias de colocación tienen sentido con reg. de long. variable
- Primer ajuste: más rápido
- Mejor ajuste: genera fragmentación interna
- Peor ajuste: genera fragmentación externa

Archivos - Operaciones

- Modificaciones
  - Consideraciones iniciales
    - Registro de long. Variable, se altera el tamaño
      - Menor, puede no importar (aunque genere fragmentación interna o externa)
      - Mayor, no cabe en el espacio
- Otros problemas
  - Agregar claves duplicadas, y luego se modifica
  - Cambiar la clave del registro (que pasa con el orden)

Clase 4

Archivos - Búsqueda

Búsqueda de información (costo)

- # de comparaciones (operaciones en memoria)
  - Se pueden mejorar con algoritmos más eficientes.

• # de accesos (operaciones en disco)

Buscar un registro

• + rápido si conocemos el NRR (directo)
• Secuencia debe buscarse desde el principio
• Trataremos de incorporar el uso de claves o llaves.

Archivos - Búsqueda

Búsqueda binaria ➔ precondiciones

• Archivo ordenado por clave
• Registros de longitud fija

Búsqueda ➔ partir el archivo a la mitad y comparar la clave,

• puedo acceder al medio por tener long. Fija
• Si N es el # de registros, la performance será del orden de Log2 N
• Se mejora la performance de la búsqueda secuencial.

Archivos ➔Clasificación

Búsqueda binaria

•acota el espacio para encontrar información
•costo ➔ mantener ordenado el archivo

Como clasificar (ordenar) un archivo

•En RAM
•Claves en RAM
•Archivos Grandes?

Archivos ➔ Clasificación

Llevar el archivo a Ram

• Eficiencia?

Llevar las claves a Ram

• Eficiencia?

Si no caben en Ram las claves

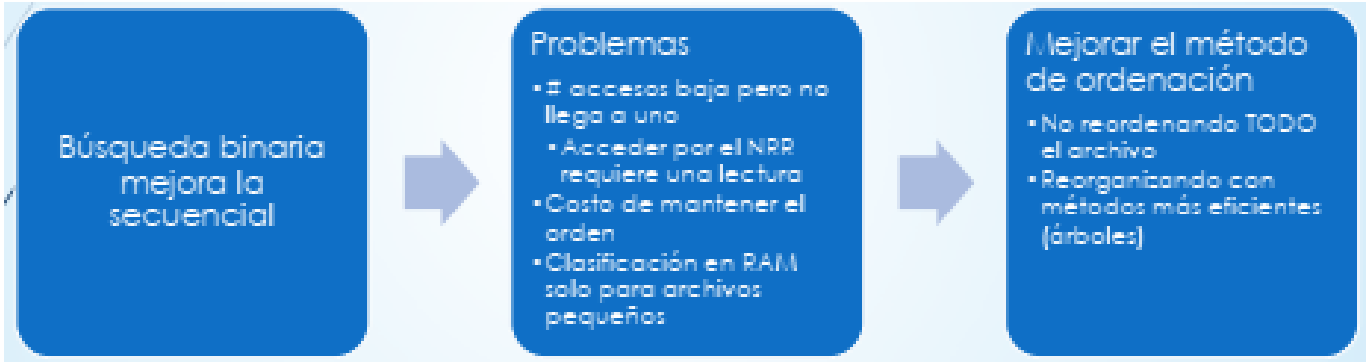
• Ordenar sobre disco?
    • Eficiencia?
• Alternativa

Archivos ➔ clasificación

Archivos demasiado grandes para caber en memoria Ram

•Partir el archivo
•Ordenar cada parte
•Juntar las partes ordenadas (merge)

Archivos ➔ Algunas conclusiones



Clase 5

Búsqueda de datos - Indices

Búsqueda de información:

• debemos minimizar el número de accesos

Secuencial (poco eficiente)

Binaria (muy costosa)

Estructuras auxiliares

Búsqueda de datos - Indices

Ejemplo:

Las últimas págs. de un libro suelen contener un índice (tabla que contiene una lista de temas y los nº de pág. Donde pueden encontrarse)

El uso de un índice es mejor alternativa que buscar un tema a lo largo del libro en forma secuencial

Otro ejemplo: encontrar libros en una biblioteca (por autor, título o tema)

• Alternativa 1: disponer 3 copias de cada libro y 3 edificios de biblioteca separados.

• Edificio1: libros clasificados por autor,

• Edificio 2: libros clasif por titulo,

• Edificio 3: libros clasif por tema (absurdo)

• Alternativa 2: usar un catálogo de tarjetas. En realidad es un conjunto de 3 índices, cada uno tiene una campo clave distinto, pero todos tienen el mismo número de catálogo como campo de referencia.

El uso de índices proporciona varios caminos de acceso a un archivo

Indices ➔ defínción

Herramienta para encontrar registros en un archivo. Consiste de un campo de llave (búsqueda) y un campo de referencia que indica donde encontrar el registro dentro del archivo de datos.

Tabla que opera con un procedimiento que acepta información acerca de ciertos valores de atributos como entrada (llave), y provee como salida, información que permite la rápida localización del registro con esos atributos.

Estructura de datos (clave, dirección) usada para decrementar el tiempo de acceso a un archivo.

Índice: equivale a índice temático de un libro

(tema, #hoja) (clave, NRR/distancia en bytes) Estructura más simple es un árbol

Característica fundamental Permite imponer orden en un archivo sin que realmente este se reacomode este se r

Indices ➔ Ejemplo

Dir. Reg.	Cía	Nº ID	Título	Compositores	Artista
32	LON	2312	Romeo y Julieta	Prokofiev	Maazel
77	RCA	2626	Cuarteto en Do...	Beethoven	Juilliard
132	WAR	23699	Touchstone	Corea	Corea
167	ANG	3795	Sinfonía Nº 9	Beethoven	Giulini
211	COL	38358	Nebraska	Springsteen	Springsteen
256	DG	18807	Sinfonía Nº 9	Beethoven	Karajan
300	MER	75016	Suite el Gallo...	Rymsky-Korsakov	Leinsdorf
353	COL	31809	Sinfonía Nº 9	Dvorak	Bernstein
396	DG	139201	Concierto para Violín	Beethoven	Ferras
422	FF	245	Good News	Sweet Honey In..	Sweet Honey In..

Llave primaria: cía grabadora + Nº de identificación de la cía

• Forma canónica: cía en mayúsculas + Nº identificación

• No se puede hacer búsqueda binaria sobre el archivo ya que tiene reg. de longitud variable (no se puede usar en NRR como medio de acceso)

Dos Archivos: índice y datos

• Se construye un índice: llave de 12 caracteres (alineada a izq. y completada con blancos) más un campo de referencia (dir. del primer byte del registro correspondiente)

• Estructura del índice: archivo ordenado de reg. de long fija (puede hacerse búsqueda binaria).

• En memoria

• Más fácil de manejar que el arch. de datos

Llave	Ref	Dir. de registro	Registro de Datos
ANG3795	167	32	LON¦2312¦Romeo y Julieta¦Prokofiev...
COL31809	353	77	RCA¦2626¦Cuartetoen Do...
COL38358	211	132	WAR¦23699¦Touchstone¦Corea...
DG139201	396	167	ANG¦3795¦Sinfonía Nº9¦Beethoven...
DG18807	256	211	COL¦38358¦Nebraska¦Springsteen...
FF245	442	256	DG¦18807¦Sinfornía Nº 9¦Beethoven...
LON2312	32	300	MER¦76016¦Suite El gallo de Oro¦Rimsky...
MER75016	300	353	COL¦31809¦Sinfornía Nº9¦Dvorak...
RCA2626	77	396	DG¦139201¦Concierto para violín¦Beethoven...
WAR23699	132	422	FF¦245¦Good News¦Sweet Honey in the....

Indices ➔ como implantarlos?

Operaciones básicas en un archivo indizado

• Índice en memoria (búsqueda binaria + rápida, comparada con archivos clasificados)

• Crear los archivos (el indice y el archivo de datos se crean vacíos, solo con registro cabecera)

• Cargar el índice en memoria (se supone que cabe, ya que es lo suficientemente pequeño. Se almacena en un arreglo)

• Reescritura del archivo de índice (cambios ➔ reescribir)

Indices ➔como implantarlos?

Agregar nuevos registros
• Implica agregar al archivo de datos y al archivo de indices
• Archivo de datos: copiar al final (se debe saber el NRR (fija) o distancia en bytes (variable) para el índice)
• Índice ordenarse con cada nuevo elemento en forma canónica (en mem.), setear el flag anterior
Eliminar un registro
• Arch. datos ➔ Cualquier técnica de las vistas para reutilizar el espacio
• Arch. índices ➔ se quita la entrada (ó se podría marcar como borrado).
Actualización de registros
• Sin modificar la clave (que pasa con el índice?)
    • Si el registro no cambia de longitud, se almacena en la misma posición física, el índice "no se toca".
    • Si el reg. cambia de longitud (se agranda) y se reubica en el arch. de datos ➔ se debe guardar la nueva posición inicial en el índice
    • Si se trata de long. Fija, no hay que hacer mas actividad

• Modificando la clave (que sucede?)
    • Se modifica el archivo de datos
    • Se debe actualizar y reorganizar el archivo de índices
    • Cómo simplificar ➔ Modificar = Eliminar + Agregar (ya vistos)

Indices ➔ Resumen
Ventajas
• Se almacena en memoria principal
• Permite búsqueda binaria
• El mantenimiento es menos costoso
Desventajas
• Si no caben en memoria RAM?
• Reescritura del archivo de índices?
• Persistencia de datos

Indices ➔ Persistencia de Datos

Indices secundarios
- No sería natural solicitar un dato por clave
- En su lugar se utiliza normalmente un campo mas fácil de recordar (ej: buscar una canción por su título o por su compositor)
- Este campo es un campo que pertenece a una llave secundaria porque puede repetirse
- Las claves secundarias se pueden repetir
- El índice secundario relaciona la llave secundaria con la llave primaria
- Acceso ➔ 1º por llave secundaria (se obtiene la clave primaria) y luego llave primaria (en índice primario)

Indice de	Compositores
Llave Secundaria	Llave Primaria
BEETHOVEN	ANG3795
BEETHOVEN	DG139201
BEETHOVEN	DG18807
BEETHOVEN	RCA2626
COREA	WAR23699
DVORAK	COL31809
PROKOFIEV	LON2312
RIMSKY-KORSAKOV	MER75016
SPRINGSTEEN	COL38358
SWEET HONEY....	FF245

Problemas: la repetición de información

• El arch. de índices se debe reacomodar con cada adición, aunque se ingrese una clave secundaria ya existente, dado que existe un 2do orden por la clave primaria.
    • Se desperdicia espacio
    • Menor posibilidad de que el índice quepa en memoria
• Misma clave varias ocurrencias, en distintos registros

Soluciones

•Arreglo: clave + vector de punteros con ocurrencias
    BEETHOVEN ANG3795 DG139201 DG18807 RCA2626
•Al agregar un nuevo reg. de una clave existente no se debe reacomodar nada-> solo reacomodar el vector de ocurrencias
•Al agregar un nuevo reg. con una clave nueva, se genera un arreglo con la clave y un elemento en el vector de punteros

Problema: elección del tamaño del vector.

•Tamaño fijo
    •Puede haber casos en que sea insuficiente
    •Puede haber casos que sobre espacio, provocando fragmentación interna
•Mejora: clave + lista de punteros con ocurrencias

NRR	Archivo de índice secundario	
0	BEETHOVEN	3
1	COREA	2
2	DVORAK	7
3	PROKOFIEV	10
4	RIMSKY-KORSAKOV	6
5	SPRINGSTEEN	4
6	SWET HONEY IN...	9

NRR	Arch de listas de llaves primarias	
0	LON2312	-1
1	RCA2626	-1
2	WAR23699	-1
3	ANG3795	8
4	COL38358	-1
5	DG18807	1
6	MER76016	-1
7	COL31809	-1
8	DG139201	5
9	FF245	-1
10	ANG36193	0

Listas invertidas:

- Archivos en los que una llave secundaria lleva a un conjunto de una o más claves primarias ➔ lista de referencias de claves primarias

- No se pierde espacio (no hay reserva)
- Si se agrega un elem. a la lista ➔ no se necesaria una reorganización completa

Organización física:
- Archivos secundarios
- Marcas o referencias

Operaciones:
- Agregar un nuevo consiste en agregar concurrencias en la lista invertida
- Idem borrar
- Modificaciones dependiendo el caso

Índices secundarios

Ventajas

•El único reacomodamiento en el arch. índice -> al agregar o cambiar un nombre

•Borrar o añadir grabaciones para un compositor->sólo cambiar el archivo de listas

•Como el reacomodamiento es a bajo costo se podría almacenar el arch. índice en mem. secundaria, liberando RAM

Desventaja

•el arch. de listas es conveniente que esté en memoria ppal. porque podría haber muchos desplazamientos en disco ➔ costoso si hay muchos índices secundarios