

Fundamentos de Organización de Datos

Archivos
Maestro - Detalle

Algorítmica clásica sobre archivos

Archivo maestro: Resume información sobre el dominio de un problema específico.

Ejemplo: *El archivo de productos de una empresa.*

Archivo detalle: Contiene movimientos realizados sobre la información almacenada en el maestro.

Ejemplo: *archivo conteniendo las ventas sobre esos productos.*

Algorítmica clásica sobre archivos

Importante: Analizar las precondiciones de cada caso particular.
Los algoritmos a desarrollar deben tener en cuenta estas precondiciones, caso contrario determina la falla de su ejecución.

Actualización de un archivo maestro con un archivo detalle - Precondiciones

- Existe un archivo maestro.
- Existe un único archivo detalle que modifica al maestro.
- Cada registro del detalle modifica a un solo registro del maestro que seguro existe.
- No todos los registros del maestro son necesariamente modificados.
- Cada elemento del maestro que se modifica, es alterado por un solo un elemento del archivo detalle.
- Ambos archivos están ordenados por igual criterio.

Ejemplo: Definición de tipos

```
type
    producto = record
        cod: string[4];
        descripcion: string[30];
        pu: real; {precio unitario}
        stock: integer;
    end;
    venta_prod = record
        cod: string[4];
        cant_vendida: integer;
    end;

    maestro = file of producto;
    detalle = file of venta_prod;
```

Ejemplo: variables y operaciones

var

```
mae: maestro;  
det: detalle;  
regm: producto;  
regd: venta_prod;
```

begin { Inicio del programa }

```
assign(mae, 'maestro.dat');  
assign(det, 'detalle.dat');  
reset(mae);  
reset(det);
```

Ejemplo: algoritmo

{Loop archivo detalle}

while not(EOF(det)) **do begin**

read(mae, regm); // Lectura archivo maestro

read(det, regd); // Lectura archivo detalle

{Se busca en el maestro el producto del detalle}

while (regm.cod <> regd.cod) **do**

read(mae, regm);

{Se modifica el stock del producto con la cantidad vendida de ese producto}

```
regm.stock := regm.stock-regd.cant_vendida;
```

{Se reubica el puntero en el maestro}

```
seek(mae, filepos(mae)-1);
```

{Se actualiza el maestro}

```
write(mae, regm);
```

```
end; // Fin while archivo detalle
```

```
close(det);
```

```
close(mae);
```

```
end.
```

Actualización de un archivo maestro con un archivo detalle

- Existe un archivo maestro.
- Existe un único archivo detalle que modifica al maestro.
- Cada registro del detalle modifica a un registro del maestro que seguro existe.
- No todos los registros del maestro son necesariamente modificados.
- Cada elemento del archivo maestro puede no ser modificado, o ser **modificado por uno o más elementos del detalle**.
- Ambos archivos están ordenados por igual criterio.

Ejemplo: Definición de tipos

```
type
    producto = record
        cod: string[4];
        descripcion: string[30];
        pu: real;
        stock: integer;
    end;
    venta_prod = record
        cod: string[4];
        cant_vendida: integer;
    end;
    detalle = file of venta_prod;
    maestro = file of producto;
```

Ejemplo: variables y operaciones

```
var
    mae: maestro;
    det: detalle;
    regm: producto;
    regd: venta_prod;
    cod_actual: string[4];
    tot_vendido: integer;

begin  {Inicio del programa}
    assign(mae, 'maestro');
    assign(det, 'detalle');
    reset(mae);
    reset(det);
```

Ejemplo: algoritmo

{Loop archivo detalle}

while not(**EOF**(det)) **do begin**

read(mae, regm); // Lectura archivo maestro

read(det, regd); // Lectura archivo detalle

{Se busca en el maestro el producto del
detalle}

while (regm.cod <> regd.cod) **do**

read(mae, regm);

{Se totaliza la cantidad vendida del detalle}

```
cod_actual := regd.cod;  
tot_vendido := 0;  
while (regd.cod = cod_actual) do begin  
    tot_vendido:=tot_vendido+regd.cant_vendida;  
    read(det, regd);  
end;
```

{Se actualiza el stock del producto con la
cantidad vendida del mismo}

```
regm.stock := regm.stock - tot_vendido;
```

```
{se reubica el puntero en el maestro}
seek(mae, filepos(mae)-1);
{se actualiza el maestro}
write(mae, regm);
end;
close(det);
close(mae);
end.
```

¿Diferencia entre este ejemplo y el anterior?

Se agrega una iteración que permite agrupar todos los registros del detalle que modificarán a un elemento del maestro.

¿Inconvenientes de esta solución?

La segunda operación **read** sobre el archivo detalle se hace sin controlar el fin de datos del mismo. Podría solucionarse agregando un **if** que permita controlar dicha operación, pero cuando finaliza la iteración interna, al retornar a la iteración principal se lee otro registro del archivo detalle, perdiendo así un registro.

Actualización de un archivo maestro con un archivo detalle

```
const valoralto = 'ZZZZ';
type str4 = string[4];
producto = record
    cod: str4;
    descripcion: string[30];
    pu: real;
    stock: integer;
end;
venta_prod = record
    cod: str4;
    cant_vendida: integer;
end;
detalle = file of venta_prod;
maestro = file of producto;
```

Ejemplo

```
var
    mae: maestro;
    det: detalle;
    total: integer;
    regm: producto;
    regd: venta_prod;
    aux: str4;

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

{programa principal}

begin

assign(mae, 'maestro') ;

assign(det, 'detalle') ;

reset(mae) ;

reset(det) ;

read(mae, regm) ;

 leer(det, regd) ;

{Se procesan todos los registros del archivo detalle}

```
while (regd.cod <> valoralto) do begin
    aux := regd.cod;
    total := 0;
```

{Se totaliza la cantidad vendida de productos iguales en el archivo de detalle}

```
while (aux = regd.cod) do begin
    total := total + regd.cant_vendida;
    leer(det, regd);
end;
```

{se busca el producto del detalle en el maestro}

while (regm.cod <> aux) **do**

read (mae, regm);

{se modifica el stock del producto con la cantidad total vendida de ese producto}

 regm.stock := regm.stock - total;

{se reubica el puntero en el maestro}

seek(mae, filepos(mae)-1);

{se actualiza el maestro}

write(mae, regm);

{se avanza en el maestro}

if (**not**(EOF(mae))) **then**

read(mae, regm);

end;

close(det);

close(mae);

end.



¿Dudas?