Actualizacion

Tipos de Actualización

POR LOTES
Varios registros de movimiento para un registro maestro.

UNITARIA
Un registro movimiento para un registro maestro.

Ficheros

Ficheros de entrada: por lo menos dos: MAESTRO Y MOVIMIENTOS.
Ficheros de salida: como mínimo uno.

Tipos de Ficheros Maestros

- Histórico
- Común o Normal

Tipos de Ficheros de Movimientos

- Actualización General o Combinado
- Actualización Parcial o Individual

Tipos de Movimientos

- Alta
- Baja
- Modificación

Algoritmo Actualización Unitaria

Utilizando ciclo incluyente

```
Accion Act_Uni es
  Algoritmo
    Abrir_Archivos
    Leer_Maestro
    Leer_Movimiento
    Mientras (Clave_Mae <> HV) o (Clave_Mov <> HV) hacer
      Si (Clave_Mae = Clave_Mov) entonces
        Proceso_Iguales
      sino
        Si (Clave_Mae < Clave_Mov) entonces
          Reg_sal:= Reg_mae
          Escribir(Arch_sal, Reg_sal)
          Leer_Maestro
        sino
          Proceso_Distintos
        Fin Si
      Fin Si
    Fin Mientras
    Cerrar_Archivos
Fin Accion

Subaccion Leer_Movimiento es
  Leer(Arch_mov, Reg_mov)
  Si FDA(Arch_mov) entonces
    Clave_mov:= HV
  Fin Si
Fin Subaccion

Subaccion Leer_Maestro es
  Leer(Arch_mae, Reg_mae)
  Si FDA(Arch_mae) entonces
    Clave_mae : = HV
  Fin Si
Fin Subaccion

Subacción Proceso_Iguales es
  Si Reg_mov.Cod_mov = 'ALTA' entonces
    Escribir('Error alta no existe')
    Reg_sal:= Reg_mae
    Escribir(Arch_sal, Reg_sal)
  Sino
    Si (Reg_mov.Cod_Mov = 'MODIF') entonces
      Proceso_Mod_Maestro
      Reg_sal : = Reg_mae
      Escribir(Arch_sal, Reg_sal)
    Sino // eliminación lógica
      Marcar_Registro
      Reg_sal:= Reg_mae
      Escribir(Arch_sal, Reg_sal)
    Fin Si
  Fin Si
  Leer_Maestro
  Leer_Movimiento
```

```
Fin Subacción.

Subaccion Proceso_Distintos es
  Si (Reg_mov.Cod_Mov = 'BAJA') entonces
    Escribir('Error baja no existe')
  Sino
    Si (Reg_mov.Cod_Mov = 'MODIF') entonces
      Escribir('Error mofificación no existe')
    Sino // Asigna campo por campo, porque Reg_sal y Reg_mov tienen distinto
formato
      Reg_sal.clave:= Reg_mov.clave
      Reg_sal.campo1:= Reg_mov.campo1
      Reg_sal.campo2:= Reg_mov.campo2
      ........
      Reg_sal.campoN:= Reg_mov.campoN
      Reg_sal.Marca_baja:= ''
      Escribir(Arch_sal, Reg_sal)
    Fin Si
  Fin Si
  Leer_Movimiento
Fin Subaccion

Subaccion Proceso_Modif_Maestro es
  Si (Reg_Mov.campo1 <> '') entonces
    Reg_mae.campo1:= Reg_mov.campo1
  Fin Si
  Si (Reg_Mov.campo2 <> '') entonces
    Reg_mae.campo2 := Reg_mov.campo2
  Fin Si
  Si (Reg_Mov.campo3 <> '') entonces
    Reg_mae.campo3 := Reg_mov.campo3
  Fin Si
    // ... y así sucesivamente para todos los campos del registro...
Fin Subaccion

Subaccion Marcar_Registro es
  Reg_mae.Marca_baja:= '*'     // en vez de asterisco, se puede asignar la fecha
del día,
                                   o cualquier otro dato, según el problema //
Fin Subaccion
```

Actualizacion Secuencial Por lotes

Consideraciones

- Utilizamos un ciclo **incluyente**.
- Los tipos de movimientos son:
  - **Alta:** Siempre el primer registro va a ser del tipo alta, y luego modificaciones.

- **Modificación:** pueden existir varias modificaciones a un maestro.
- **Baja:** son bajas logicas. Solo el ultimo movimiento puede ser una baja logica.

- No existen altas o bajas entre las modificaciones.

Ambiente

El ambiente esta formado por 2 archivos con la siguiente estructura:

```
Formato_Clave = Registro
  clave1: ...
  clave2: ...
  ...
  claven: ...
fin registro

formato_Maestro = Registro
  clave: formato_clave
  campo1: ...
  campo2: ...
  ...
  campon: ...
  Marca_baja: ...
fin registro

formato_Movimiento = Registro
  clave: formato_clave
  campo1: ...
  campo2: ...
  ...
  campon: ...
  cod_mov: ("A", "M", "B")
fin registro

mae, mae_sal: archivo de formato_Maestro ordenado por clave
reg_sal, reg_mae, aux: formato_Maestro

mov: archivo de formato_movimiento ordenado por clave
reg_mov: formato_movimiento
```

Algoritmo

```
Abrir_Archivos
Leer_Maestro
Leer_Movimiento
MIENTRAS (reg_mov.clave <> High_Value) o (reg_mae.clave <> High_Value) HACER
```

```
  SI Creg_mae.clave < Creg_mov.clave ENTONCES
    //Maestro sin Movimiento
    Reg_sal : = Reg_mae
    ESCRIBIR(mae_sal, Reg_sal)
    Leer_Maestro
  SINO
    SI Creg_mae.clave = Creg_mov.clave Movimiento
      aux : = reg_mae
      MIENTRAS (Creg_mae.clave = Creg_mov.clave) HACER
        Proceso_Movim
        Leer_Movimiento
      Fin Mientras
      reg_sal : = Aux
      ESCRIBIR(mae_sal, reg_sal)
      Leer_Maestro
    SINO
      // Movimiento sin Maestro ~ 1 Alta y 0-1 Modific. y/o Bajas
      // Asigna campo por campo, porque Aux y Reg_mov tienen distinto formato
      Aux.clave  := Reg_mov.clave
      Aux.campo1 := Reg_mov.campo3
      Aux.campo2 := Reg_mov.campo4
      ........
      Aux.campon := Reg_mov.campon
      Aux.Marca_baja := ""
      Leer_Movimiento
      MIENTRAS (Clave_Aux = Clave_Mov) HACER
        Proceso_Movim
        Leer_Movimiento
      FMientras
      reg_sal : = aux
      ESCRIBIR(mae_sal, reg_sal)
    FIN SI
  FIN SI
FIN MIENTRAS

CERRAR(mae)
CERRAR(mae_sal)
CERRAR(mov)
```

Subacciones

```
Subacción Leer_Movimiento es
  LEER(mov, Reg_mov)
  SI FDA(mov) ENTONCES
    reg_mov.clave := High_value
  FSI;
FAcción

Subacción Leer_Maestro es
  LEER(mae, reg_mae)
```

```
  SI FDA(mae) ENTONCES
    reg_mae.clave : = High_value
  FSI;
FAcción

Subacción Proceso_Movim es
  //Comparar la información del Registro de Movimientos y, de acuerdo a los
valores que
  //tengan, y alterar los contenidos del Registro Auxiliar (Aux).
  SI reg_mov.Cod_Mov = "M" ENTONCES
    //Modificación
    Proceso_modif_maestro
  SINO
    SI reg_mov.Cod_Mov = "B" ENTONCES
      //eliminación lógica
      Marcar_registro
    Fsi
  Fsi
FAcción

Subacción Proceso_modif_maestro es
  Si Reg_Mov.campo1 <> "" entonces
    // No se actualizan los campos claves.
    Aux.campo1 := Reg_mov.campo1
  fsi;
  Si Reg_Mov.campo2 <> "" entonces
    Aux.campo2 := Reg_mov.campo2
  fsi;
  ..............................
  //... y así sucesivamente para todos los campos del registro...
  ..............................
  Si Reg_Mov.campon <> "" entonces
    Aux.campon := Reg_mov.campon
  fsi;
FAcción.

Subacción Marcar_registro es
  //en vez de asterisco, se puede asignar la fecha del día,
  //o cualquier otro dato, según el problema
  Aux.Marca_baja:= "*"
FAcción.
```

Modificar

- Que pasa si se pide que se tengan en cuenta los siguientes errores:
- Las altas no siempre estan al principio.
- Las bajas no siempre estan al final.

Actualizacion Indexada

Consideraciones

- Archivo maestro indexado por clave
- Se ingresan movimientos por terminal

Ambiente

El ambiente esta formado por 1 solo archivos con la siguiente estructura:

```
formato_mae = Registro
  clave: formato_clave
  campo1: ...
  campo2: ...
  ...
  campon: ...
  Marca_baja: ...
fin registro

arch_mae: Archivo de formato_mae indexado por clave.
reg_mae: formato

cod_mov: AN(1)

valido = ('A', 'B', 'M')

clave: formato_clave
```

Algoritmo

```
Abrir E/S (arch)

Escribir (‘Por favor ingrese la clave a procesar y el código de movimiento (A:
incorporaciones, B: bajas, M: modificaciones) Para finalizar ingrese cualquier
otra letra.’)

Leer(clave, cod_mov)

MIENTRAS (cod_mov en valido) HACER
  reg_mae.clave := clave
  LEER (arch_mae, reg_mae)

  SI no existe ENTONCES
    SI cod_mov = 'B' ENTONCES
```

```
      ESCRIBIR('Error baja no existe')
    SINO
      SI cod_mov = 'M' ENTONCES
        ESCRIBIR('Error modificación no existe')
      SINO
        // Ingresar por teclado los datos correspondientes a la nueva clave
        LEER(reg_mae.campo1)
        LEER(reg_mae.campo2)
        ...
        LEER(reg_mae.campon)
        ESCRIBIR(arch_mae, reg_mae)
      FSI
    FSI
  SINO
    SI cod_mov = 'A' ENTONCES
      ESCRIBIR('Error clave existe, alta no es posible')
    SINO
      SI cod_mov = 'M' ENTONCES
        // Ingresar por teclado los datos que se desean modificar
        LEER(campo1)
        SI campo1 <> '' ENTONCES
          reg_mae.campo1 := campo1
        FSI
        LEER(campo2)
        SI campo2 <> '' ENTONCES
          reg_mae.campo2 := campo2
        FSI
        ...
        LEER(campon)
        SI campon <> '' ENTONCES
          reg_mae.campon := campon
        FSI
        RE-ESCRIBIR(arch_mae, reg_mae)
      SINO
      // Baja lógica
        reg_mae.Marca_baja := "*"
        RE-ESCRIBIR(arch_mae, Reg_mae)
      // Baja Física
        BORRAR(arch_mae, reg_mae)
      FSI
    FSI
  FSI


  ESCRIBIR ('Por favor ingrese la clave a procesar y el código de movimiento (A: 
incorporaciones, B: bajas, M: odificaciones) Para finalizar ingrese cualquier 
otra letra.')

  LEER(clave, cod_mov)

FIN MIENTRAS
```

```
CERRAR(arch_mae)
```