

Conceptos de Bases de Datos

Agustin Murray

March 18, 2024

Contents

1	Conceptos generales de bases de datos	3
1.1	Conceptos Basicos	3
1.2	Origenes de las BD	3
1.3	Gestores de Bases de Datos	3
2	Archivos, estructuras y operaciones basicas	5
2.1	Definicion	5
2.2	Aspectos fisicos	5
2.3	Niveles de vision	5
2.4	Organizacion interna de los datos	6
2.5	Acceso a informacion contenida en los archivos	6
2.6	Operaciones basicas sobre archivos	6
2.6.1	Definicion de archivos	6
2.6.2	Correspondencia archivo logico - archivo fisico	7
2.6.3	Apertura y creacion de archivos	7
2.6.4	Cierre de archivos	7
2.6.5	Lectura y escritura de archivos	7
2.6.6	Operaciones adicionales con archivos	8
3	Algoritmica clasica sobre archivos	9
3.1	Objetivo	9
3.2	Actualizacion de un archivo maestro con un archivo detalle (I)	9
3.3	Actualizacion de un archivo maestro con un archivo detalle (II)	10
3.4	Actualizacion de un archivo maestro con N archivos detalle	11
3.5	Proceso de generacion de un nuevo archivo a partir de otros existentes. Merge	12
3.5.1	Primer ejemplo	12
3.5.2	Segundo ejemplo	13
3.6	Corte de control	14

1 Conceptos generales de bases de datos

1.1 Conceptos Basicos

- Se considera **Base de Datos**, a una coleccion o conjunto de datos interrelacionados con un proposito especifico vinculado a la resolucion de un problema del mundo real.
- Cualquier informacion dispuesta de manera adecuada para su tratamiento por una computadora.
- Una coleccion de archivos diseñados para servir a multiples aplicaciones.

1.2 Origenes de las BD

En los origenes de las **BD**, las unidades de almacenamiento de gran volumen eran muy lentas, entonces se buscaba reducir el acceso a los mismos. Para esto, era necesario repetir datos en distintos puestos de trabajo y al final del dia actualizar los mismos en todas las unidades para evitar problemas de perdida/modificacion de informacion.

Con el tiempo, la tecnologia fue avanzando y los sistemas de informacion evolucionaron. Se logra integrar aplicaciones, interrelacionar archivos y eliminar la redundancia de datos.

1.3 Gestores de Bases de Datos

Un **Sistema de Gestion de Bases de Datos (SGBD)** consiste en un conjunto de programas necesarios para acceder ya administrar una BD.

Actualmente, cualquier sistema de software necesita interactuar con informacion almacenada en una BD y para ello requiere del soporte de un SGBD.

Un SGBD posee dos tipos diferentes de lenguajes: uno para especificar el esquema de una BD, y el otro para la manipulacion de los datos.

La definicion del esquema de una BD implica:

- Diseno de la estructura que tendra efectivamente la BD.
- Describir los datos, la semantica asociada y las restricciones de consistencia.

Para ello se utiliza un lenguaje especial, llamado **Lenguaje de Definicion de datos (LDD)**. El resultado de compilar lo escrito con el LDD es un archivo llamado **Diccionario de Datos**. Un **Diccionario de Datos** es un archivo con metadatos, es decir, datos acerca de los datos.

Los objetivos mas relevantes de un SGBD son:

- **controlar la concurrencia:** varios usuarios pueden acceder a la misma informacion en un mismo periodo de tiempo. Si el acceso es para consulta, no hay inconvenientes, pero si mas de un usuario quiere actualizar el mismo dato a la vez, se puede llegar a un estado de inconsistencia que, con la supervision del SGBD, se puede evitar.
- **Tener control centralizado:** tanto de los datos como de los programa que acceden a los datos.

- **Facilitar el acceso a los datos:** dado que provee un lenguaje de consulta para recuperacion rapida de informacion.
- **Proveer seguridad para imponer restricciones de acceso:** se debe definir explicitamente quienes son los usuarios autorizados a acceder a la BD.
- **Mantener la integridad de los datos:** esto implica que los datos incluidos en la BD respeten las condiciones establecidas al definir la estructura de la BD y que, ante una falla del sistema, se posea la capacidad de restauracion a la situacion previa.

2 Archivos, estructuras y operaciones basicas

2.1 Definicion

Un **archivo** es una coleccion de registros semejantes, guardados en dispositivos de almacenamiento secundario de una computadora.

Los archivos se caracterizan por el crecimiento y las modificaciones que se efectuan sobre estos. El crecimiento indica la incorporacion de nuevos elementos, y las modificaciones involucran alterar datos contenidos en el archivo, o quitarlos.

2.2 Aspectos fisicos

Almacenamiento primario (RAM):

- Capacidad de almacenamiento limitada.
- Volatil.
- Alto costo.
- Acceso rapido (orden de los nanosegundos)

Almacenamiento secundario (cintas y discos):

- Alta capacidad de almacenamiento.
- No volatil.
- Menor costo que el primario.
- Acceso lento a comparacion del primario (orden de los milisegundos). Se busca optimizar este aspecto:
 1. Búsqueda de un unico dato: Obtencion en un intento o en pocos.
 2. Búsqueda de varios datos: obtencion de todos de una sola vez.
- **Cintas:** acceso secuencial, economico, estable en diferentes condiciones ambientales, facil de transportar.
- **Discos:** acceso directo, se almacenan datos en sectores.

2.3 Niveles de vision

- **Archivo fisico:** es el archivo residente en la memoria secundaria y es administrado (ubicacion, tipo de operaciones disponibles) por el sistema operativo.
- **Archivo logico:** es el archivo utilizado desde el algoritmo. Cuando el algoritmo necesita operar con un archivo, genera una conexion con el sistema operativo, el cual sera el responsable de la administracion. Esta accion se denomina independencia fisica.

2.4 Organizacion interna de los datos

- **Secuencia de bytes:** Se determina como unidad mas pequeña de L/E al byte. No se puede determinar facilmente el comienzo y el final de cada dato por lo que generalmente son archivos de texto.
- **Campos:** Se determina como unidad mas pequeña de L/E al campo. El campo es un item de datos elemental y se caracteriza por su tipo de dato y su tamaño.
- **Registros:** Se determina como unidad mas pequeña de L/E al registro. El registro es un conjunto de campos agrupados que definen un elemento del archivo. Los campos internos a un registro deben estar logicamente relacionados, como para ser tratados como una unidad.

2.5 Acceso a informacion contenida en los archivos

Basicamente, se pueden definir tres formas de acceder a los datos de un archivo:

- **Secuencial:** el acceso a cada elemento de datos se realiza luego de haber accedido a su inmediato anterior. El recorrido es, entonces, desde el primero hasta el ultimo de los elementos, siguiendo el orden fisico de estos.
- **Secuencial indizado:** el acceso a los elementos de un archivo se realiza teniendo presente algun tipo de organizacion previa, sin tener en cuenta el orden fisico.
- **Directo:** es posible recuperar un elemento de dato de un archivo con un solo acceso, conociendo sus características, mas alla de que exista un orden fisico o logico predeterminado.

2.6 Operaciones basicas sobre archivos

Para poder operar con archivos, son necesarias una serie de operaciones elementales disponibles en todos los lenguajes de programacion que utilicen archivos de datos. Estas operaciones incluyen:

- La definicion del archivo logico.
- La definicion de la forma de trabajo del archivo (creacion inicial, utilizacion).
- La administracion de datos (L/E info).

2.6.1 Definicion de archivos

Como cualquier otro tipo de datos, los archivos necesitan ser definidos. Se reserva la palabra clave **file** para indicar la definicion del archivo.

```
Var archivo_logico: file of tipo_de_dato;
```

Otra opcion para definir archivos se presenta a continuacion:

```
Type archivo = file of tipo_de_dato;  
Var archivo_logico: archivo;
```

2.6.2 Correspondencia archivo logico - archivo fisico

Se debe indicar que el archivo logico utilizado por el algoritmo se corresponde con el archivo fisico adminstrado por el sistema operativo. La sentencia encargada de hacer esta correspondencia es:

```
Assign (nombre_logico , nombre_fisico );
```

2.6.3 Apertura y creacion de archivos

Hasta el momento, se ha detallado como definir un archivo y se ha esctablecido la relacion con el nombre fisico. Para operar con un archivos desde un algoritmo, se debe realizar la apertura.

- La operacin **rewrite** indica que el archivo va a ser creado y, por lo tanto, la unica operacion valida sobre el mismo es escribir informacion.
- La operacino **reset** indica que el archivo ya existe y, por lo tanto, las operaciones validas sobre el mismo son L/E de informacion.

```
rewrite(nombre_logico );  
reset(nombre_logico );
```

2.6.4 Cierre de archivos

Para mantener valida la marca de end of file (EOF) se cierra el archivo de la siguiente forma:

```
close (nombre_logico );
```

2.6.5 Lectura y escritura de archivos

Para leer o escribir informacion en un archivo, las instrucciones son:

```
read(nombre_logico , var_dato );  
write(nombre_logico , var_dato );
```

Buffers de memoria: Las lecturas y escrituras desde o hacia un archivo se realizan sobre buffers. Se denomina buffer a una memoria intermedia (ubicada en RAM) entre un archivo y un programa, donde los datos residen provisoriamente hasta ser almacenados definitivamente en la memoria secundaria, o dodnde los datos residen una vez recuperados de dicha memoria secundaria. La razon de esto es la mejora de performance al trabajar con mayor frecuencia en memoria principal.

La operacion read lee desde un buffer y, en caso de no contar con informacion, el sistema operativo realiza automaticamente una operacion input, trayendo mas informacion al buffer. La diferencia radica en que cada operacion input transfiere desde el disco una serie de registros. De esta forma, cada determinada cantidad de instrucciones read, se realiza una operacion input.

De forma similar procede la operacion write; en este caso, se escribe en el buffer, y si no se cuenta con espacio suficiente, se descarga el buffer a disco por medio de una operacion output, dejandolo nuevamente vacio.

2.6.6 Operaciones adicionales con archivos

- Control fin de datos:

`eof(nombre_logico);`

La funcion retornara verdadero si el puntero del archivo referencia e EOF, y falso en caso contrario.

- Control de tamaño del archivo:

`filesize(nombre_logico);`

- Control de posicion de trabajo dentro del archivo:

`Filepos(nombre_logico);`

- Ubicacion fisica en alguna posicion del archivo:

`seek(nombre_logico , posicion);`

- Truncamiento de un archivo a partir de la posicion acutal:

`truncate(nombre_logico);`

3 Algoritmica clasica sobre archivos

3.1 Objetivo

Desarrollar algoritmos considerados clasicos en la operatoria de archivos secuenciales. Estos algoritmos se resumen en tres tipos: de actualizacion, merge y corte de control.

El primer caso, con todas sus variantes, permite introducir problemas donde se actualiza el contenido de un archivo resumen o "maestro", a partir de un conjunto de archivos con datos vinculados a este archivo maestro.

En el segundo caso, se dispone de la informacion distribuida en varios archivos que se reune para generar un nuevo archivo, producto de la union de los anteriores.

Por ultimo, el corte de control, muy presente en la operatoria de BD, determina situaciones done, a partir de informacion contenida en archivos, es necesario generar reportes que resuman el contenido, con un formato especial.

Se denomina archivo maestro al archivo que resume informacion sobre un dominio de problema especifico. Ejemplo: el archivo de productos de una empresa que contiene el stock actual de cada producto. Por otra parte, se denomina archivo detalle al archivo que contiene novedades o movimientos realizados sobre la informacion almacenada en el maestro. Ejemplo: el archivo con todas las ventas de los productos de la empresa realizadas en un dia particular.

3.2 Actualizacion de un archivo maestro con un archivo detalle (I)

Presenta la variante mas simple del proceso de actualizacion. Las precondiciones del problema son las siguientes:

- Existe un archivo maestro.
- Existe un unico archivo detalle que modifica al maestro.
- Cada registro del detalle modifica a un registro del maestro. Esto significa que solamente aparecern datos en el detalle que se correspondan con datos del maestro. Se descarta la posibilidad de generar altas en ese archivo.
- No todos los registros del maestro son necesariamente modificados.
- Cada elemento del maestro que se modifica es alterado por uno y solo un elemento del archivo detalle.
- Ambos archivos estan ordenados por igual criterio. Esta precondicion, considerada esencial, se debe a que hasta el momento se trabaja ocn archivos de datos de acuerdo con su orden fisico.

begin

```
assign(mael, 'maestro');  
assign(det1, 'detalle');  
reset(mael);  
reset(det1);  
while not eof(det1) do begin  
  read(mael, regm);
```

```

    read(det1, regd);
    while(regm.cod <> regd.cod) do read(mael, regm);
    regm.stock := regm.stockk - regd.cant_vendida;
    seek(mael, filepos(mael)-1);
    write(mael, regm);
end;
close(det1);
close(mael);
end.

```

3.3 Actualizacion de un archivo maestro con un archivo de-talle (II)

Este es otro caso, levemente diferente del anterior; solo se modifica una precondition del problema y hace, de esta forma, variar el algoritmo resolutorio.

- Cada elemento del archivo maestro puede no ser modificado, o ser modificado por uno o mas elementos del detalle (hay registros repetidos).

```

program ejemplo_3_3;
const valoralto='9999';
type str4 = string[4]

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

begin
    assign(mael, 'maestro');
    assign(det1, 'detalle');
    reset(mael);

\include{1.tex}
    reset(det1);
    read(mael, regm);
    read(det1, regd);

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

    while(aux = regd.cod) do
begin
        total := total + regd.cant_vendida;
        leer(det1, regd);
    end;

```

```

while (regm.cod < aux) do read(mael, regm);

regm.cant := regm.cant - total;

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

write(mael, regm);

if not eof(mael) then read(mael, regm);
end;
close(det1);
close(mael);
end.

```

El procedimiento de lectura, denominado leer, es el responsable de realizar el read correspondiente sobre el archivo, en caso de que este tuviera mas datos. En caso de alcanzar el fin de archivo, se asigna a la variable dato.cod, por la cual el archivo esta ordenado, un valor imposible de alcanzar en condiciones normales de trabajo. Este valor indicara que el puntero del archivo ha llegado a la marca de fin.

3.4 Actualizacion de un archivo maestro con N archivos detalle

Bajo las mismas consignas del ejemplo anteriores, se plantea un proceso de actualizacion donde, ahora, la cantidad de archivos detalle se lleva a N (siendo $N > 1$) y el resto de las precondiciones son las mismas.

El ejemplo 3.4 presenta la resolucion de un algoritmo de actualizacion a partir de tres archivos detalle. Para ello, se agrega un nuevo procedimiento, denominado minimo, que actua como filtro. El objetivo de este proceso a partir de la informacion recibida es retornar el elemento mas pequeño de acuerdo con el criterio de ordenamiento del problema. El objetivo del procedimiento minimo es determinar el menor de los tres elementos recibidos de cada archivo y leer otro registro del archivo desde donde provenia ese elemento.

```

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

procedure minimo(var r1, r2, r3, min:venta_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(det, r2);
  end

```

```

    end
  esle
  begin

\include{1.tex}
    min := r3;
    leer(det3,r3);
    end;
end;

begin
  {asignacion y apertura de archivos correspondientes}

  while(min.cod<>valoralto) do
  begin
    aux:=min.cod;
    total_vendido:=0;
    while(aux=min.cod) do
    begin
      total_vendido:=total_vendido+min.cantvendida;
      minimo(regd1,regd2,regd3,min);
    end;
    while(regm.cod<>min.cod) do read(mael,regm);

    regm.cant:=regm.cant-total;
    seek(mael, filepos(mael)-1);
    write(mael,regm);
    if not eof(mael) then read(mael,regm);
  end;
  close(det1);
  close(det2);
  close(det3);
  close(mael);
end.

```

3.5 Proceso de generacion de un nuevo archivo a partir de otros existentes. Merge

3.5.1 Primer ejemplo

El primer ejemplo plantea un problema muy simple. Las precondiciones son las siguiente:

- Se tiene informacion en tres archivos detalle.
- Esta informacion se encuentra ordenada por el mismo criterio en cada caso.
- La informacion es disjunta; esto significa que un elemento puede aparecer una sola oportunidad en todo el problema.

```

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

  minimo(regd1, regd2, regd3, min);
  while(min.codigo < valoralto) do
    begin
      write(mael, min);
      minimo(regd1, regd2, regd3, min);
    end;
  close(det1);
  close(det2);
  close(det3);
  close(mael);
end.

```

3.5.2 Segundo ejemplo

Como segundo ejemplo se presenta un problema similar, pero ahora los elementos se pueden repetir dentro de los archivos detalle, modificando de esta forma la tercera precondition del ejemplo anterior. El resto de las precondiciones permanecen inalteradas.

```

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

  minimo(regd1, regd2, regd3, min);

  while(min.codigo < valoralto) do
    begin
      codprod:=min.codigo;
      cantotal:=0;
      while(codprod=min.codigo)
        begin
          cantotal:=cantotal+min.cant;
          minimo(regd1, regd2, regd3, min);
        end;
      write(mael, min);
    end;
  close(det1);
  close(det2);
  close(det3);
  close(mael);
end.

```

3.6 Corte de control

Se denomina corte de control al proceso mediante el cual la informacion de un archivo es presentada en forma organizada de acuerdo con la estructura que tiene el archivo. Suponga que se almacena en un archivo la informacion de ventas de una cadena de electrodomesticos. Dichas ventas han sido efectuadas por los vendedores de cada sucursal de cada ciudad de cada provincia del pais. Luego, es necesario informar al gerente de ventas de la empresa el total de ventas producidas.

Deben tenerse en cuenta las siguientes precondiciones:

- El archivo se encuentra ordenado por provincia, ciudad, sucursal y vendedor.
- Se debe informar el total vendido en cada sucursal, ciudad y provincia, asi como el total final.
- En diferentes provicinas pueden existir ciudades con el mismo nombre, o en diferentes ciudades pueden existir sucursales con igual denominacion.

begin

{ asignaciones y aperturas correspondientes. }

leer (archivo , reg);

total:=0;

while(reg.Provincia \Diamond valoralto) **do**

begin

writeln('Provincia:-' , reg.Provincia);

prov:=reg.Provincia;

totprov:=0;

while(prov=reg.Provincia) **do**

begin

writeln('Ciudad:-' , reg.Ciudad);

ciudad:=reg.Ciudad;

totciudad:=0;

while(prov=reg.Provincia) **and** (Ciudad=reg.Ciudad) **do**

begin

writeln('Sucursa:-' , reg.sucursal);

sucursal:=reg.Sucursal;

totsuc:=0;

while((prov=reg.Provincia) **and**

(Ciudad=reg.Ciudad) **and** Sucursal=reg.Sucursal) **do**

begin

write('Vendedor:-' , reg.Vendedor);

writeln(reg.MontoVenta);

totsuc:=totsuc+reg.MontoVenta;

leer (archivo , reg);

end;

writeln('Total-sucursal' , totsuc);

totciudad:=totciudad+totsuc;

end;

writeln('Total-ciudad' , totciudad);

totprov:=totprov+totciudad;

```
    end;  
    writeln( 'Total-provincia ', totprov );  
    total:=total+totprov;  
end;  
writeln( 'Total-empresa ', total );  
close( archivo );  
end.
```