**Conceptos basicos:**

**Que es una Base de datos?**

* Es una coleccion de datos relacionados.
* Coleccion de archivos diseñados para servir a multiples aplicaciones
* Un dato representa hechos conocidos que pueden registrarse y que tienen un resultado implicito.

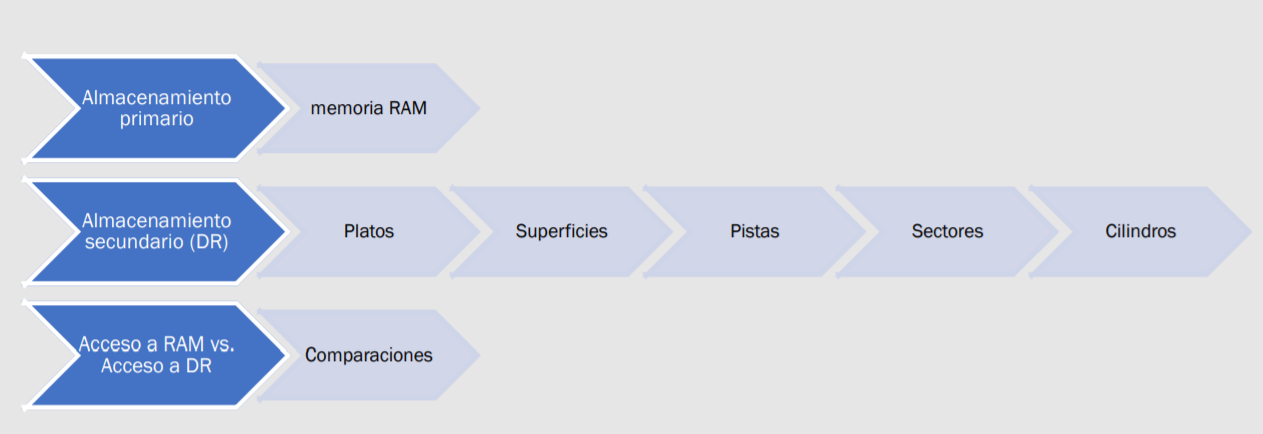
**Propiedades implicitas de una BD**

* Una BD representa algunos aspectos del mundo real, a veces denominado Universo de Discurso.
* Una BD se diseña, construye y completa de datos para un proposito especifico. Esta destinada a un grupo de usuarios concretos y tiene algunas aplicaciones preconcebidas en las cuales estan interesados los usuarios
* Una BD es una coleccion coherente de datos con significados inherentes. Un conjunto aleatorio de datos no puede considerarse una BD. O sea, los datos deben tener cierta logica.
* Una BD esta sustentada fisicamente en archivos en dispositivos de almacenamiento persistente de datos

**Archivos**

Definiciones:

* Coleccion de registros guardados en almacenamiento secundario
* Coleccion de datos almacenados en dispositivos secundarios de memoria
* Coleccion de registros que abarcan entidades con un aspecto comun y originadas para algun proposito particular

**Repaso de conceptos de Hardware**

**¿Como se organiza un archivo?**

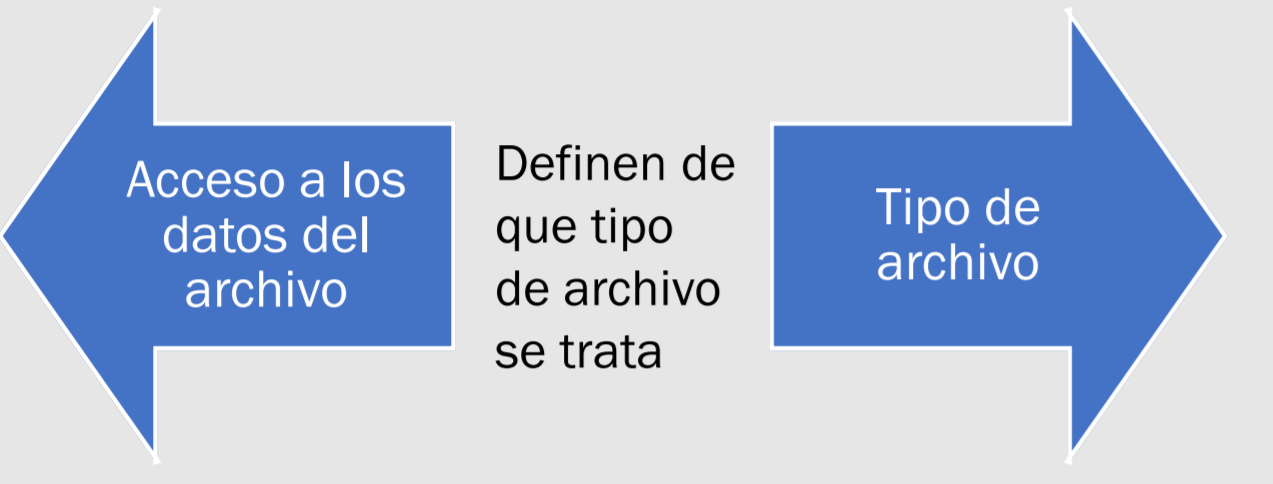
* Secuencia de bytes
  + Archivos de texto
  + Se leen o recuperan caracteres.
  + No hay formato previo.
  + Una palabra se termina por un conjunto de caracteres que termina con blanco. Pero eso es una “convencion”
* Tipos primitivos
* Conjunto de registros
* Hay estructura.
* Los registros pueden estar conformados por campos

**Archivos → organizacion**

Registro: Conjunto de campos agrupados que definen un elemento del archivo.

Campo: Unidad mas pequeña, logicamente significativa de un archivo

**Archivos**



**Formas de acceso a archivos**

Secuencial Fisico:

* acceso a los registros uno tras otro y en el orden fisico en el que estan guardados

Secuencial indizado (logico):

* acceso a los registros de acuerdo al orden establecido por otra estructura
* Ej: una guia telefonica, o indice tematico del un libro

Directo:

* se accede a un registro determinado sin necesidad de haber accedido a los predecesores

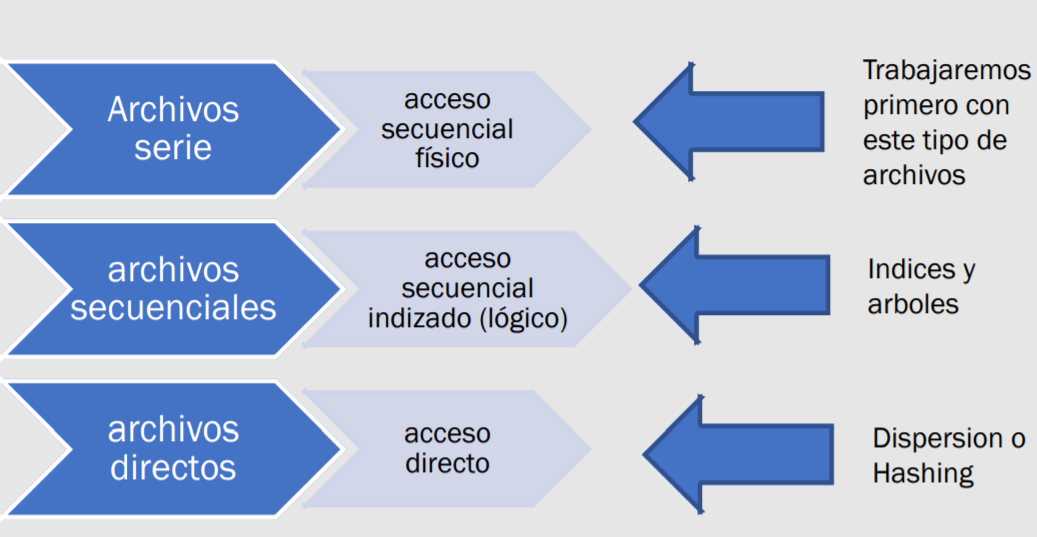
**Tipos de archivos**

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

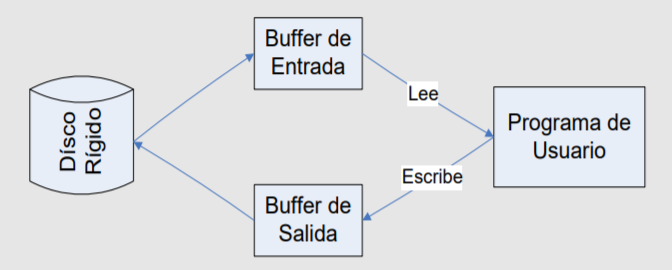
Resumiendo



**Algunos conceptos**

**Buffer**

* Memoria intermedia entre un archivo y un programa, donde los datos residen provisoriamente hasta ser almacenados definitivamente en memoria secundaria o donde los datos residen una vez recuperados de dicha memoria secundaria
* Los buffers ocupan lugar en RAM
* SO encargado de manipular los buffers
* ¿Como trabaja? Buffer de entrada y Buffer de salida

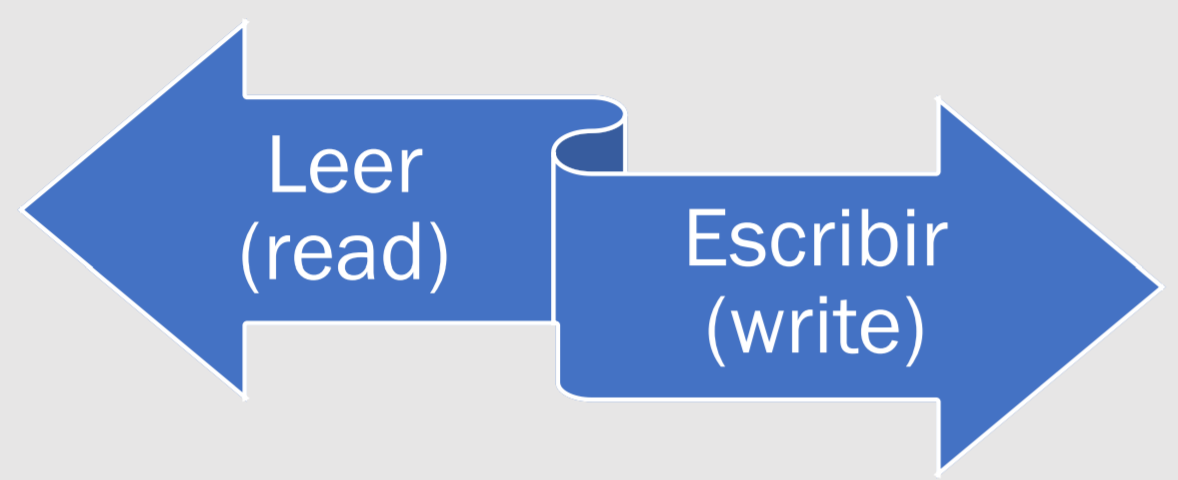
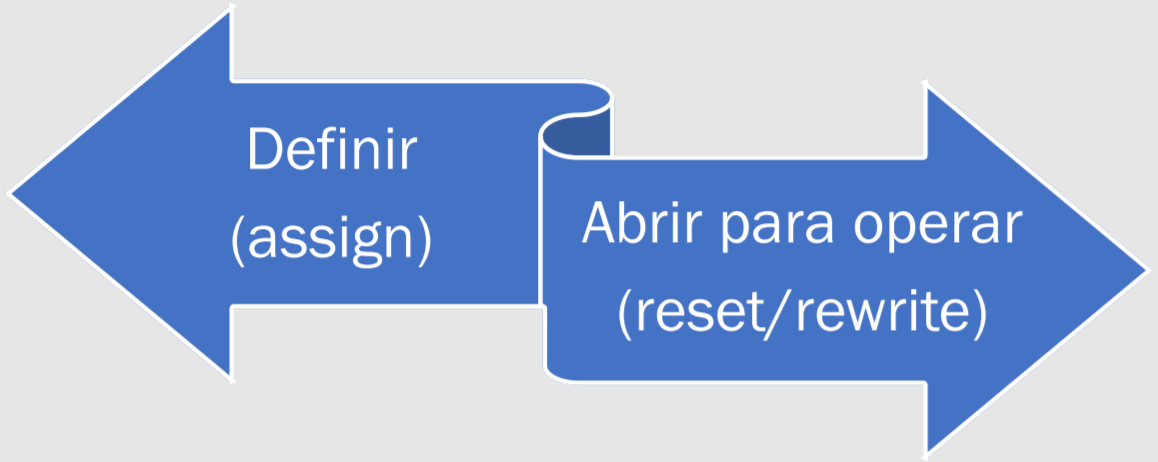


Archivos en los lenguajes de programacion

Archivos Fisicos: •como aparecen en el disco rigido •a cargo del sistema operativo

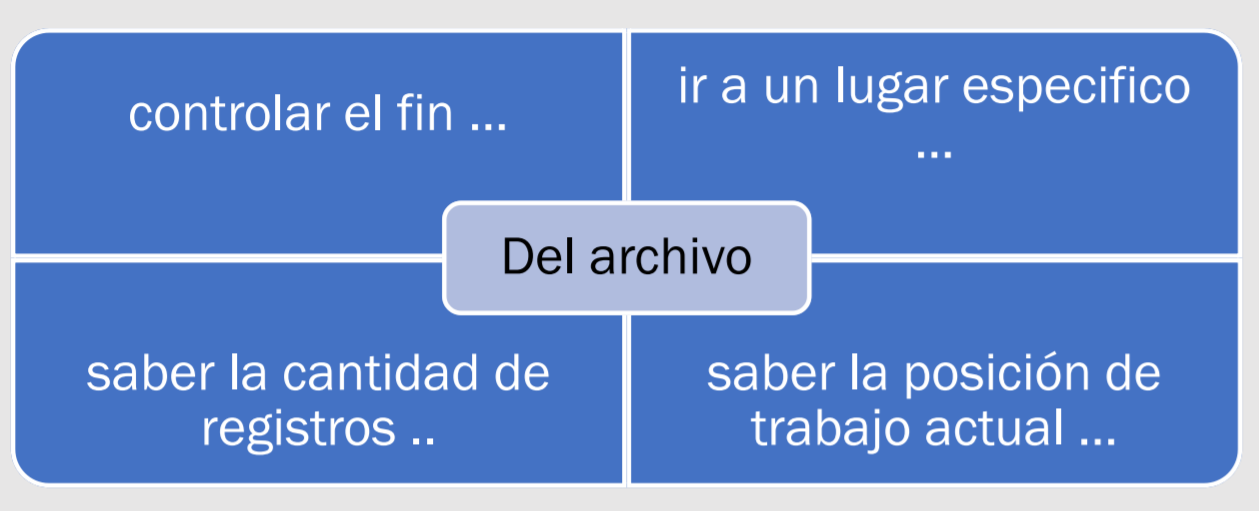
Archivos logicos: •como se manipulan desde el lenguaje de programacion

**Archivos: Operaciones iniciales Archivos: Operaciones basicas**



Assign = definir, reset/rewrite= abrir para operar, read = leer y write = escribir.

**Archivos: operaciones adicionales**



Archivos: Definir/declarar (notacion Pascal)

■ Variable

– Var archivo: file of Tipo\_de\_dato;

■ Tipo

– Type archivo: file of Tipo\_de\_dato;

– Var arch: archivo

**Relacion con el sistema operativo**

– Se tiene que establecer la correspondencia entre el nombre fisico y nombre logico



**Archivos operar**

Rewrite (nombre\_logico);

• De solo escritura (creacion )

Reset (nombre\_logico);

• Lectura Escritura (apertura)

Close(nombre\_logico);

• Cierre de archivo

• Esta instruccion indica que no se va a trabajar mas con el archivo. Significa poner una marca de EOF (end of file) al final del mismo.

**Archivos Operaciones basicas**

Read(nombre\_logico, variable);

Write(nombre\_logico, variable);

•Estas operaciones leen y/o escriben sobre los buffers relaciona-dos a los archivos

•No se realizan directamente sobre el DR

(En ambos casos la variable debe ser del mismo tipo que los elementos que se declararon como parte del archivo)



**Archivos: Operaciones Adicionales**

EOF( nombre\_logico); (funcion)

•Fin de archivo

FileSize(nombre\_logico); (funcion)

•Tamaño del archivo

FilePos( nombre\_logico); (funcion)

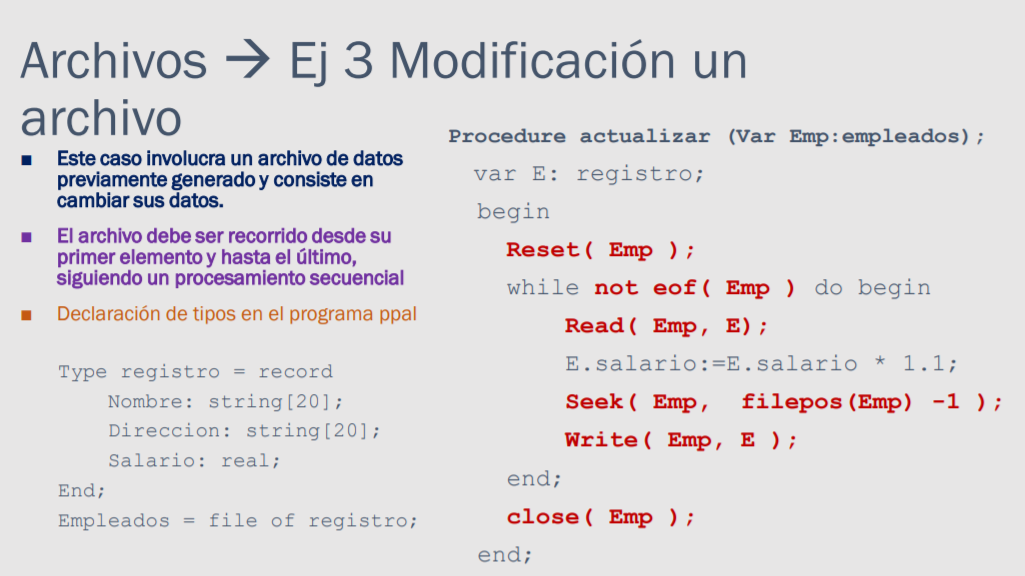
•Posicion dentro del archivo

Seek( nombre\_logico, posicion); (Procedimiento)

•Ir a una posicion del archivo

•La posicion se cuenta siempre desde el comienzo del archivo

•El primer lugar es el cero.



------------------------------------------------------LEER TEORIAS-----------------------------------------------------

Archivos 🡪 Actualizacion 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 informacion que se utilizara para modificar el contenido del archivo maestro.
* En general
  + Un maestro
  + N detalles.
* Consideraciones del proceso (precondiciones)
  + Ambos archivos (maestro y detalle) estan 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 🡪 Un Maestro Un detalle

Archivos 🡪 Un Maestro N detalle

Archivos 🡪 Corte de control

* El problema consiste en la generacion de reportes
  + Es un problema clasico en el manejo de BD.
  + Si bien los DBMS lo manejan diferente, veremos la algoritmica clasica de los mismos
  + Precondiciones
    - El archivo se encuentra ordenado por provincia, partido y ciudad

Archivos 🡪 Merge

* Involucra archivos con contenido similar, el cual debe resumirse en un unico archivo.
* Precondiciones:
  + Todos los archivos detalle tienen igual estructura
  + Todos estan ordenados por igual criterio

Archivos 🡪 Merge N archivos con repeticion

* Precondiciones
  + Similar al anterior
  + Cada vendedor puede realizar varias ventas diarias

------------------------------------------------------------------------------------------------------------------------------

**Archivos 🡪 Introduccion**

* La memoria primaria (RAM) es rapida y de simple acceso, pero su uso tiene algunas desventajas respecto al almacenamiento secundario:
* Capacidad limitada
* Mayor costo
* Es volatil

**Almacenamiento secundario necesita mas 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 informacion esta organizada en archivos**

* Archivo: coleccion de bytes que representa informacion

Archivo Fisico:

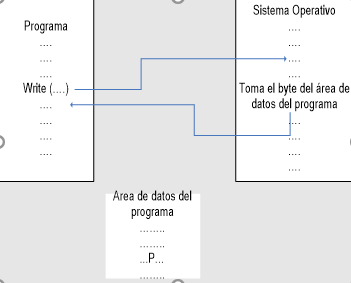
* Archivo que existe en almacenamiento secundario
* Es el archivo tal como lo conoce el S.O. y que aparece en su directorio de archivos

Archivo logico:

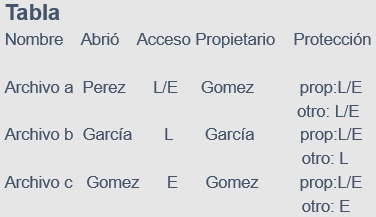
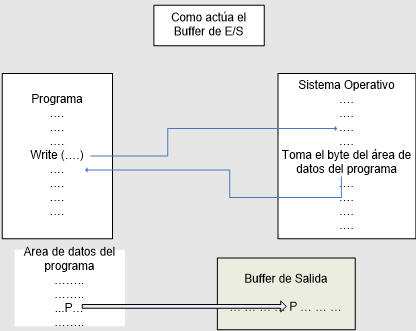
* Es el archivo, visto por el programa
* Permite a un programa describir las operaciones a efectuarse en un archivo,
* No se sabe cual archivo fisico real se utiliza o donde esta ubicado

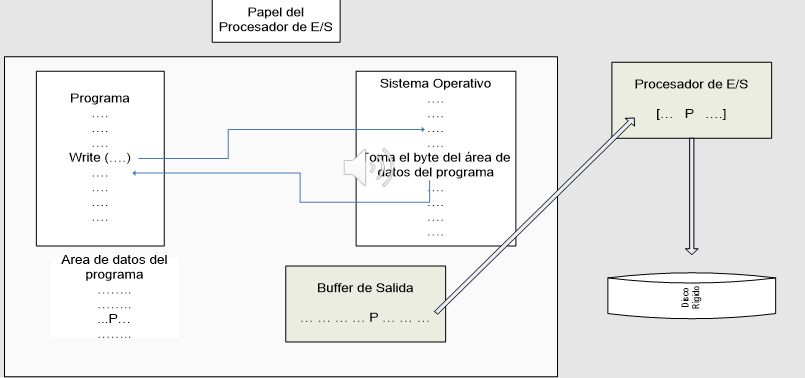
Archivos – Viaje de un byte

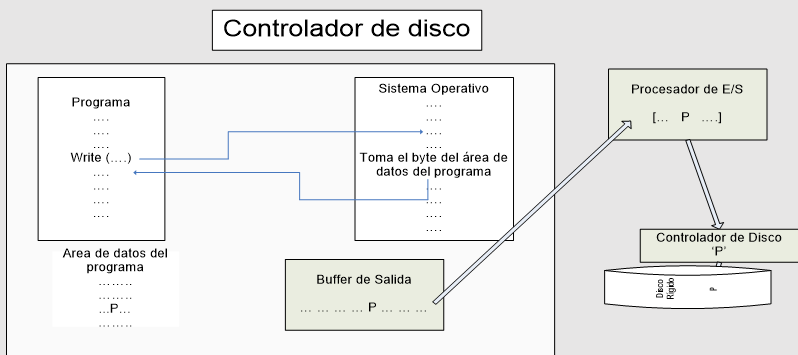
* **Viaje de un byte No es sencillo**
  + *Escribir un dato en un archivo*
    1. Write ( archivo, variable) ciclos para escribir
  + *Quienes estan involucrados*
    1. Administrador de archivos
    2. Buffer de E/S
    3. Procesador de E/S
    4. Controlador de disco
* Administrador de archivos: conjunto de programas del S.O. (capas de procedimientos) que tratan aspectos relacionados con archivos y dispositivos de E/S
* En Capas Superiores: aspectos logicos de datos (tabla)
  + Establecer si las caracteristicas del archivo son compatibles con la operacion deseada (1)
* En Capas Inferiores: aspectos fisicos (FAT)
  + Determinar donde se guarda el dato (cilindro, superficie, sector) (2)
  + Si el sector esta ubicado en RAM se utiliza, caso contrario debe traerse previamente. (3)
* **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 transmision desde o hacia almacenamiento externo. Independiente de la CPU. (3)
* **Controlador de disco**: encargado de controlar la operacion de disco.
* Colocarse en la pista
* Colocarse en el sector
* Transferencia a disco
* Que sucede cuando un programa escribe un byte en disco?
  + *Operacion*
    1. Write(……)
  + *Veamos los elementos que se involucran en esta simple operacion*
  + *Supongamos que se desea agregar un byte que representa el caracter ‘P’ almacenado en una variable c de tipo caracter, en un archivo denominado TEXTO que se encuentra en algun lugar del disco rigido.*



* **Capas del protocolo de transmision 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 caracteristicas
* El **Adm**. obtiene de la FAT la ubicacion fisica del sector del archivo donde se guardara el byte.
* El **Adm** se asegura que el sector del archivo esta 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 esta el byte en RAM y en que parte del disco debera almacenarse)
* El **procesador de E/S** encuentra el momento para transmitir el dato a disco, la CPU se libera
* El **procesador de E/S** envia el dato al **controlador de disco** (con la direccion de escritura)
* El **controlador** prepara la escritura y transfiere el dato bit por bit en la superficie del disco.





* **Archivos como Secuencia de bytes**
  + No se puede determinar facilmente comienzo y fin de cada dato.
  + Ejemplo: archivos de texto
* **Archivos estructurados**
  + **Registros** 
    - **Longitud fija o variable**
  + **Campos**
    - **Longitud fija o variable**
* **Campos**
* Unidad logicamente significativa mas pequeña de un archivo. *Permite separar la informacion*
* *Pueden ser:*
  + *Longitud fija*
    - *Cuales son?*
  + *Longitud variable*
    - *Como determinar su longitud?*
* Campos de long. Variable
* Identidad de campos: variantes, pro y contras.
  + **Longitud predecible** (long. Fija), desperdicio de espacio, si el tamaño es pequeño al agrandarlo se podria desperdiciar mas espacio)
  + **Indicador de longitud** (al ppio de cada campo)
  + **Delimitador al final de cada campo** (caracter especial no usado como dato)
* **Registros**
* Organizacion 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 direccion del byte de inicio de cada registro)
  + **Delimitador** (caracter especial no usado como dato)
* **Long. Predecible de registros**
* **Estudio de casos: ventajas y desventajas**
* **Clave**
* Se concibe al Registro como la cantidad de info. que se lee o escribe
* **Objetivo:** extraer solo un registro especifico en vez del archivo completo
* Es conveniente identificar una registro con una llave o clave que se base en el contenido del mismo
* Permite la identificacion del registro
* Deben permitir generar orden en el archivo por ese criterio
* **Univoca / Primaria:**
  + **Identifican un elemento particular dentro de un archivo**
* **Secundaria**
  + **Reconocen un conjunto de elementos con igual valor**

Archivos 🡪 Claves

* **Forma canonica**: forma estandar para una llave, puede derivarse a partir de reglas bien definidas.
* Representacion unica para la llave, ajustada a la regla
  + Ej: llave solo con letras mayusculas y sin espacios al final.
* Al introducir un registro nuevo:
  + 1ro se forma una llave canonica para ese registro
  + 2do se la busca en el archivo. Si ya existe, y es univoca 🡪 no se puede ingresar

**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**
  1. mejora el acceso a disco,
  2. no varian las comparaciones.
* **Acceso directo** 
  1. Permite acceder a un registro preciso
  2. Requiere una sola lectura para traer el dato [ O(1) ].
  3. Debe necesariamente conocerse el lugar donde comienza el registro requerido
* **Numero relativo de registro (NRR):**
  1. Indica la posicion relativa con respecto al principio del archivo
  2. 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 solo cuando se necesitan pocos registros especificos, pero este metodo NO siempre es el mas apropiado para la extraccion de info.
* Ej. generar cheques de pago a partir de un archivo de registros de empleados.
  1. Como todos los reg. se deben procesar 🡺 es mas rapido y sencillo leer registro a registro desde el ppio. hasta el final, y NO calcular la posicion en cada caso para acceder directamente.

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
  + **Estaticos** -> pocos cambios
    - Puede actualizarse en procesamiento por lotes
    - No necesita de estructuras adicionales para agilizar los cambios
  + **Volatiles** -> sometido a operaciones frecuentes:
    - Agregar / Borrar / Actualizar
    - Su organizacion debe facilitar cambios rapidos
    - Necesita estructuras adicionales para mejorar los tiempos de acceso
* Altas
* Bajas
* Modificaciones
* Consultas
* Como influye registros de long. Fija y variable

Archivos 🡪 eliminacion

* Eliminar registros de un archivo
* **Baja fisica**
* **Baja logica**
  + Cuales son las diferencias?
  + Cuales las ventajas y desventajas?
* **Eliminar**
* Cualquier estrategia de eliminacion 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 eliminacion facilmente.**
* **Como reutilizar el espacio de registros eliminados ?**
* Los programas que usan archivos deben incluir cierta logica para ignorar los registros eliminados

Archivos 🡪 eliminacion

* Recuperar el espacio
* La forma mas simple es copiar todo en un nuevo archivo a excepcion de los registros eliminados 🡺 **Baja Fisica**
* Frecuencia
  1. Tiempo (depende del dominio de aplicacion)
  2. Ante la necesidad de espacio
* Veremos el analisis de recuperacion dinamica del almacenamiento

**Aprovechamiento de espacio**

* 1. **Reg. Longitud fija** 🡺 **es necesario garantizar:**
     + Marca especiales: reg. borrados 🡺**Baja Logica**
  2. **Registros de longitud variable** 🡪 **los nuevos elementos deben “caber” en el lugar**

Recuperacion del espacio para su reutilizacion cuando se agreguen registros

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

**Aprovechamiento de espacio (reg. long. fija)**

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

**Aprovechamiento de espacio (reg. long. fija)**

* **Recuperacion de espacio con Lista o pilas (header)**
  1. Ej : en el encabezado estara NRR 4, el archivo tendra
  2. **alfa beta delta \* 6 gamma \* -1 epsilon**
* Se borra beta, como inicial quedara 2
  1. **alfa \* 4 delta \* 6 gamma \* -1 epsilon**
* Si se quiere agregar un elemento el programa solo debe chequear el header y desde ahi obtiene la direccion del primero. Agrego omega , como ppio queda 4 nuevamente
  1. **alfa omega delta \* 6 gamma \* -1 epsilon**
* **Recuperacion de espacio con reg. de longitud variable**
  1. Marca de borrado al igual que en reg. de long. fija (ej:\*)
  2. El problema de los registros de longitud variable esta en que no se puede colocar en cualquier lugar, para poder ponerlo debe caber, necesariamente.
  3. Lista . No se puede usar NRR como enlace. Se utiliza un campo binario que explicitamente indica en enlace (conviene que indique el tamaño).
  4. Cada registro indica en su inicio la cant. de bytes.
* **Recuperacion de espacio con reg. de Longitud variable**
  1. Reutilizacion: buscar el registro borrado de tamaño adecuado (lo suficientemente grande).
  2. Como se necesita buscar, no se puede organizar la lista de disponibles como una pila.
  3. El tamaño “adecuado” del primer registro borrado a reutilizar 🡪 origina Fragmentacion

**Aprovechamiento de espacio** 🡪**Fragmentacion**

* **Interna:** ocurre cuando se desperdicia espacio en un registro, se le asigna el lugar pero no lo ocupa totalmente.
  1. Ocurre, en general, con reg. long. Fija.
  2. Reg.long. Variable evitan el problema
  3. Solucion -> el “residuo” una vez ocupado el espacio libre, pasa a ser un nuevo reg. Libre. Si este es muy chico (no se podra ocupar) 🡺 fragmentacion externa
* **Externa:** ocurre cuando el espacio que no se usa es demasiado pequeño como para ocuparse. Soluciones:
  1. Unir espacios libres pequeños adyacentes para generar un espacio disponible mayor (unir los huecos en el espacio de almacenamiento)
  2. Minimizar la fragmentacion, eligiendo el espacio mas adecuado en cada caso.
* Estrategias de colocacion en registros de longitud variable:
  1. Primer ajuste
  2. Mejor ajuste
  3. Peor ajuste

**Primer ajuste:** se selecciona la primer entrada de la lista de disponibles, que pueda almacenar al registro, y se le asigna al mismo.

* Minimiza la busqueda
* No se preocupa por la exactitud del ajuste

**Mejor ajuste:** elige la entrada que mas se aproxime al tamaño del registro y se le asigna completa.

* Exige busqueda

**Peor ajuste:** selecciona la entrada mas grande para el registro, y se le asigna solo el espacio necesario, el resto queda libre para otro registro

**Conclusiones**

* Las estrategias de colocacion tienen sentido con reg. de long. variable
* Primer ajuste: mas rapido
* Mejor ajuste: genera fragmentacion interna
* Peor ajuste: genera fragmentacion externa

**Archivos - Operaciones**

**Modificaciones**

* + *Consideraciones iniciales*
    - Registro de long. Variable, se altera el tamaño
      * *Menor, puede no importar (aunque genere fragmentacion 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)

**------------------------------------------------------------------------------------------------------------------------------**

**Archivos – Busqueda**

**Busqueda de informacion (costo)**

* # de comparaciones (operaciones en memoria)
  + Se pueden mejorar con algoritmos mas eficientes.
* # de accesos (operaciones en disco)

**Buscar un registro**

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

**Busqueda binaria 🡪 precondiciones**

* Archivo ordenado por clave
* Registros de longitud fija

**Busqueda 🡪 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 sera del orden de Log2 N
* Se mejora la performance de la busqueda secuencial.

**Archivos 🡪 Clasificacion**

**Busqueda binaria**

* acota el espacio para encontrar informacion
* costo 🡪 mantener ordenado el archivo

**Como clasificar (ordenar) un archivo**

* En RAM
* Claves en RAM
* Archivos Grandes?

Archivos demasiado grandes para caber en memoria Ram

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

Archivos 🡪 Algunas conclusiones

Busqueda binaria mejora la secuencial

* Problemas
  1. # accesos baja pero no llega a uno
     1. Acceder por el NRR requiere una lectura
  2. Costo de mantener el orden
  3. Clasificacion en RAM solo para archivos pequeños
* Mejorar el metodo de ordenacion
  1. No reordenando TODO el archivo
  2. Reorganizando con metodos mas eficientes (arboles)

------------------------------------------------------------------------------------------------------------------------------

Agenda

Indices

* Definicion
* Operaciones basicas

**Busqueda de datos - Indices**

Busqueda de informacion:

* debemos minimizar el numero de accesos

Secuencial (poco eficiente)

Binaria (muy costosa)

Estructuras auxiliares

Ejemplo

* Las ultimas pags. de un libro suelen contener un indice (tabla que contiene una lista de temas y los nº de pag. donde pueden encontrarse)
* El uso de un indice es mejor alternativa que buscar un tema a lo largo del libro en forma secuencial

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

* **Alternativa 1:** disponer 3 copias de cada libro y 3 edificios de biblioteca separados.
  1. Edificio1: libros clasificados por autor,
  2. Edificio 2: libros clasif por titulo,
  3. Edificio 3: libros clasif por tema **(absurdo)**
* **Alternativa 2:** usar un catalogo de tarjetas. En realidad es un conjunto de 3 indices, cada uno tiene una campo clave distinto, pero todos tienen el mismo numero de catalogo como campo de referencia.

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

Indices 🡪 definicion

* **Herramienta** para encontrar registros en un archivo. Consiste de un *campo de llave* (busqueda) y un *campo de referencia* que indica donde encontrar el registro dentro del archivo de datos.
* **Tabla** que opera con un procedimiento que acepta informacion acerca de ciertos valores de atributos como entrada (*llave*), y provee como salida, informacion que permite la rapida localizacion del registro con esos atributos.
* **Estructura de datos** (*clave, direccion*) usada para decrementar el tiempo de acceso a un archivo.

**Indice:** equivale a indice tematico de un libro

* (tema, #hoja)
* (clave, NRR/distancia en bytes)
* **Estructura mas simple es un arbol**

Caracteristica fundamental

* **Permite imponer orden en un archivo sin que realmente este se reacomode**



**Llave primaria: cia grabadora + Nº de identificacion de la cia**

* Forma canonica: cia en mayusculas + Nº identificacion
* No se puede hacer busqueda binaria sobre el archivo ya que tiene reg. de longitud variable (no se puede usar en NRR como medio de acceso)

**Dos Archivos: indice y datos**

* Se construye un indice: llave de 12 caracteres (alineada a izq. y completada con blancos) mas un campo de referencia (dir. del primer byte del registro correspondiente)
* Estructura del indice: archivo ordenado de reg. de long fija (puede hacerse busqueda binaria).
* En memoria
* Mas facil de manejar que el arch. de datos



Operaciones basicas en un archivo indizado

* Crear los archivos (el indice y el archivo de datos se crean vacios, solo con registro cabecera)
* Cargar el indice en memoria (se supone que cabe, ya que es lo suficientemente pequeño. Se almacena en un arreglo)
* Reescritura del archivo de indice (cambios 🡪 reescribir)

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 indice)
* Indice ordenarse con cada nuevo elemento en forma canonica (en mem.), setear el flag anterior

Eliminar un registro

* Arch. datos 🡺 Cualquier tecnica de las vistas para reutilizar el espacio
* Arch. indices 🡺 se quita la entrada (o se podria marcar como borrado).

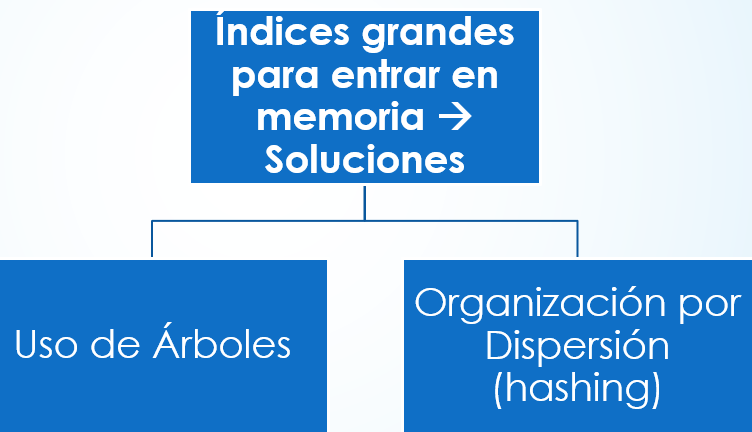
Actualizacion de registros

* Sin modificar la clave (que pasa con el indice?)
  1. Si el registro no cambia de longitud, se almacena en la misma posicion fisica, el indice “no se toca”.
  2. Si el reg. cambia de longitud (se agranda) y se reubica en el arch. de datos 🡺 se debe guardar la nueva posicion inicial en el indice
  3. Si se trata de long. Fija, no hay que hacer mas actividad
* Modificando la clave (que sucede?)
  1. Se modifica el archivo de datos
  2. Se debe actualizar y reorganizar el archivo de indices
  3. Como simplificar 🡺 Modificar = Eliminar + Agregar (ya vistos)

Indices 🡪 Resumen

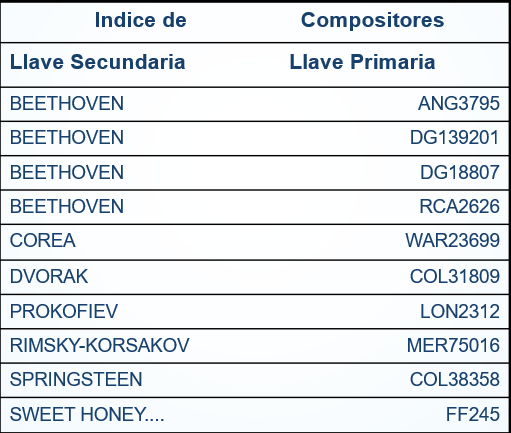
* Ventajas
  1. Se almacena en memoria principal
  2. Permite busqueda binaria
  3. El mantenimiento es menos costoso
* Desventajas
  1. Si no caben en memoria RAM?
  2. Reescritura del archivo de indices?
  3. Persistencia de datos

Indices 🡪 Persistencia de Datos



**Indices Secundarios**

* No seria natural solicitar un dato por clave
* En su lugar se utiliza normalmente un campo mas facil de recordar ( ej: buscar una cancion por su titulo 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 indice secundario relaciona la llave secundaria con la llave primaria
* Acceso 🡺 1º por llave secundaria (se obtiene la clave primaria) y luego llave primaria (en indice primario)



**Problemas: la repeticion de informacion**

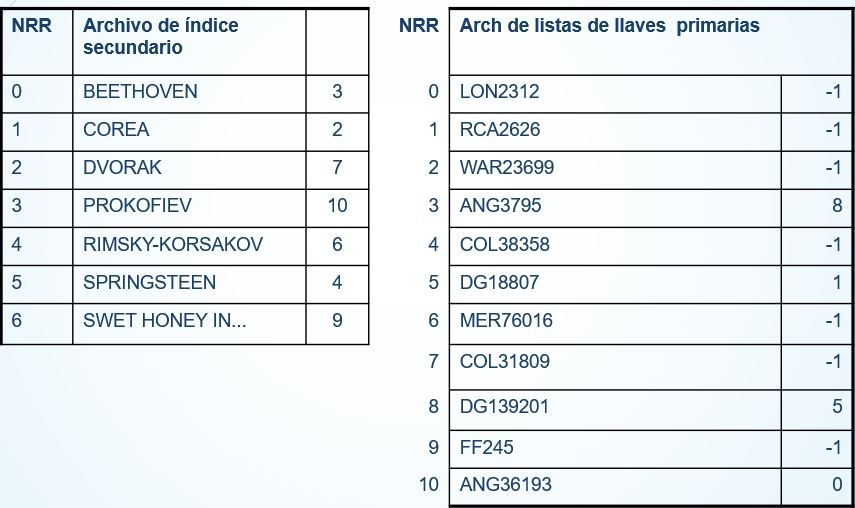
* El arch. de indices se debe reacomodar con cada adicion:
  + Puedo agregar una clave secundaria nueva
  + Puedo agregar una clave secundaria que ya existe
* Misma clave varias ocurrencias, en distintos registros

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: eleccion del tamaño del vector.

* Tamaño fijo
  1. Puede haber casos en que sea insuficiente
  2. Puede haber casos que sobre espacio, provocando fragmentacion interna
* Mejora: clave + lista de punteros con ocurrencias



**Listas invertidas:**

* Archivos en los que una llave secundaria lleva a un conjunto de una o mas 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 reorganizacion completa
* Organizacion fisica:
  + Archivos secundarios
  + Marcas o referencias
* Operaciones
  + Agregar un nuevo consiste en agregar concurrencias en la lista invertida
  + Idem borrar
  + Modificaciones dependiendo el caso

**Ventajas**

* El unico reacomodamiento en el arch. indice -> al agregar o cambiar un nombre
* Borrar o añadir grabaciones para un compositor->solo cambiar el archivo de listas
* Como el reacomodamiento es a bajo costo se podria almacenar el arch. indice en mem. secundaria, liberando RAM

**Desventaja**

* el arch. de listas es conveniente que este en memoria ppal. porque podria haber muchos desplazamientos en disco 🡺 costoso si hay muchos indices secundarios