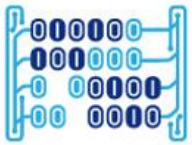


MACC
Matemáticas Aplicadas y
Ciencias de la Computación

Almacenamiento

Manejo de Bases de Datos

Juan F. Pérez

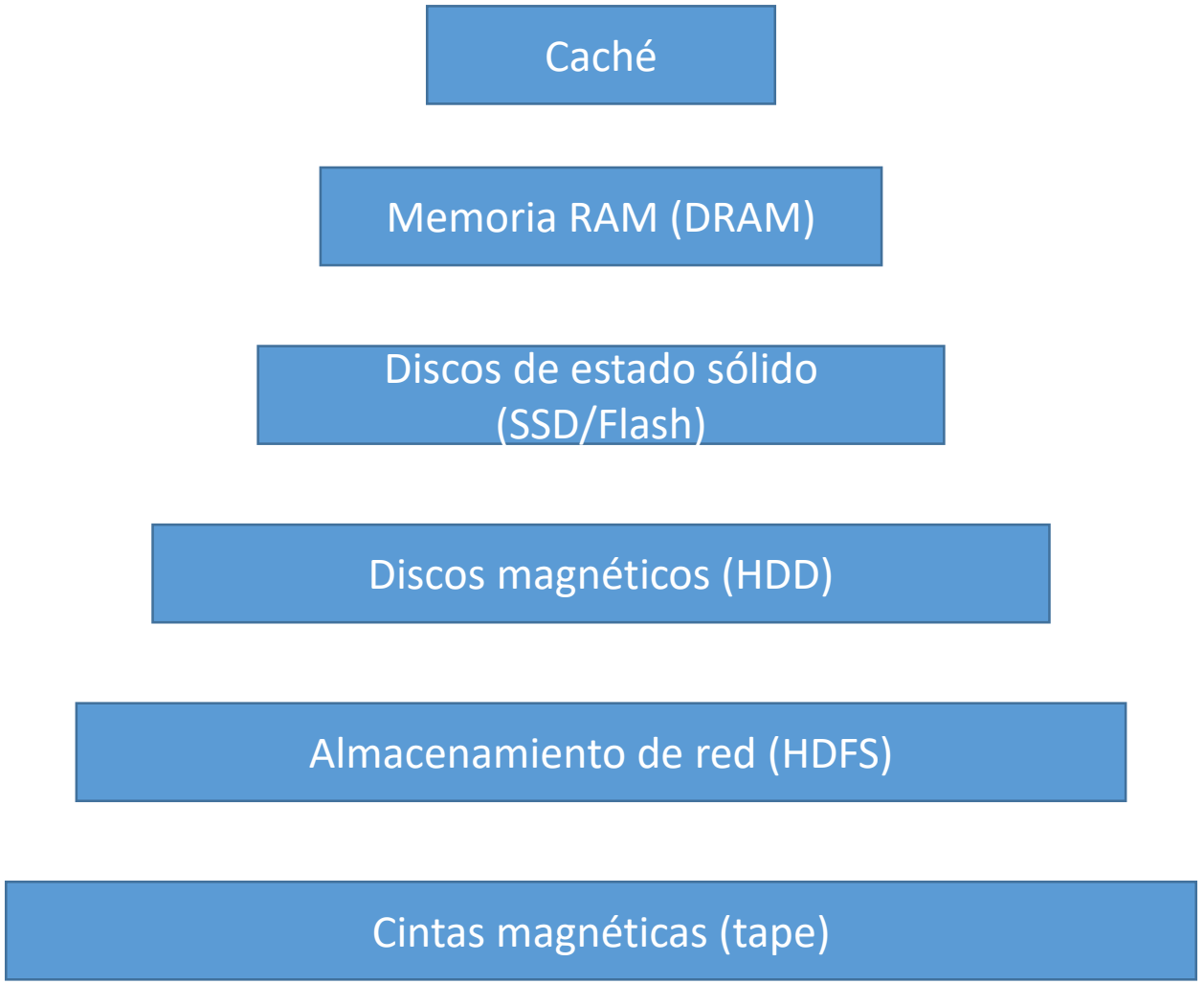
Departamento de Matemáticas Aplicadas

y Ciencias de la Computación

2019 - 1

Jerarquía de Almacenamiento

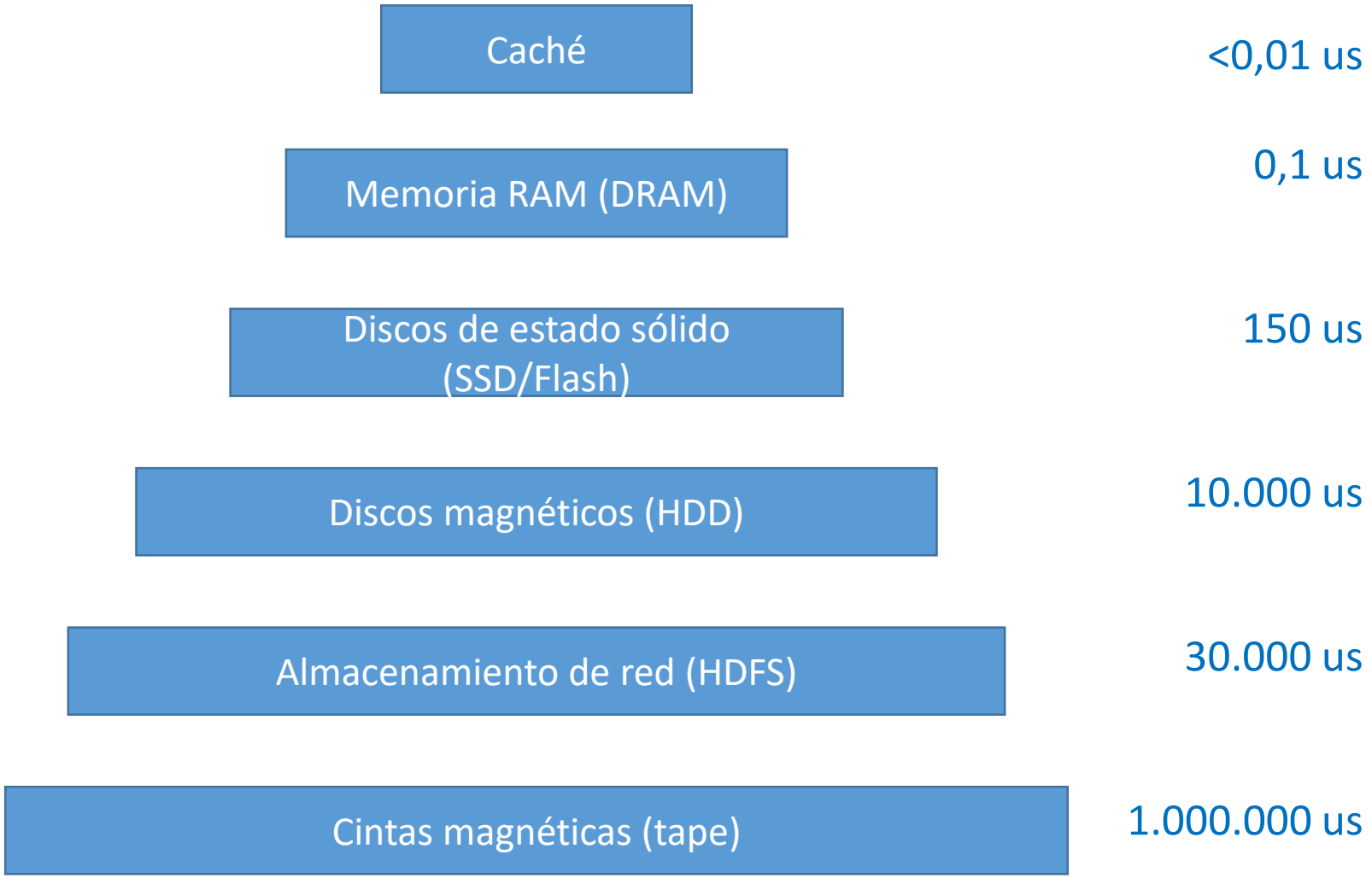
Más rápido
Más pequeño
Más costoso



Más lento
Más grande
Más barato

Jerarquía de Almacenamiento

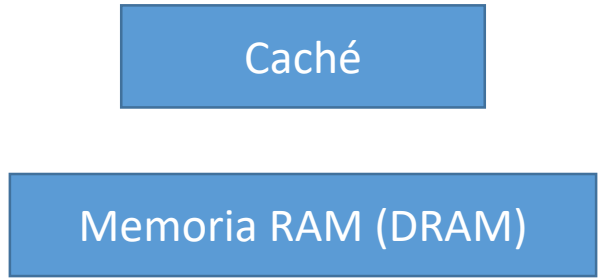
Más rápido
Más pequeño
Más costoso



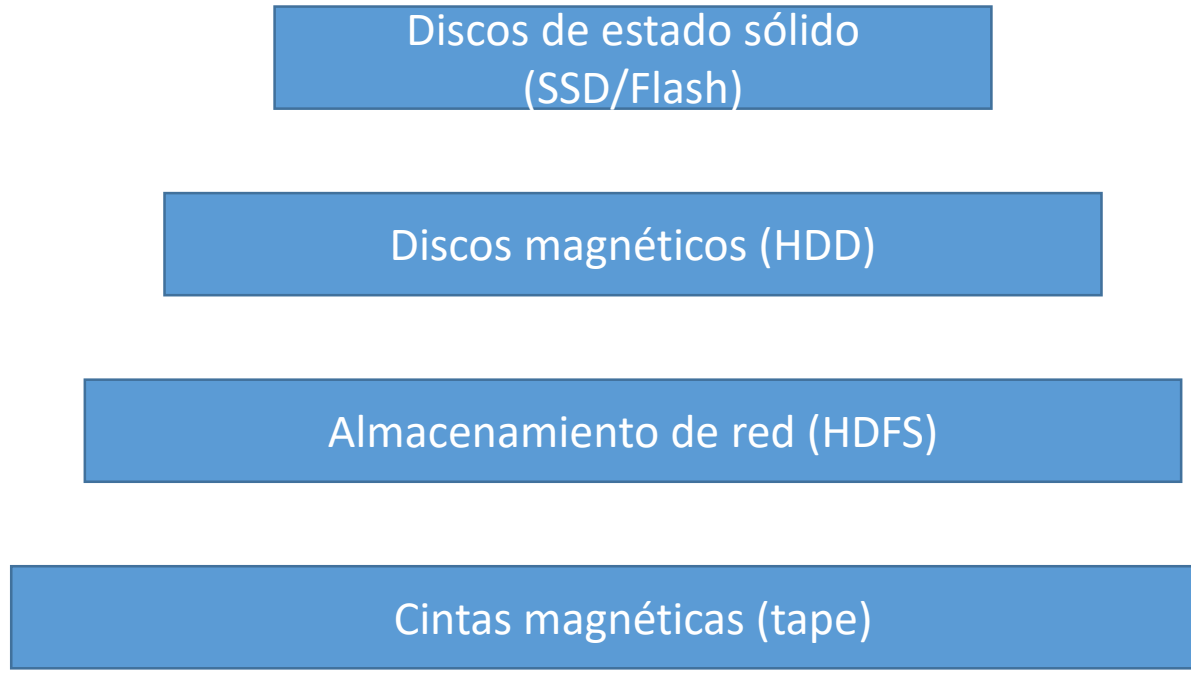
Más lento
Más grande
Más barato

Jerarquía de Almacenamiento

Volátil
Acceso aleatorio (random access)
Acceso por bytes



No Volátil
Acceso secuencial
Acceso por bloques



Jerarquía de Almacenamiento

Volátil
Acceso aleatorio (random access)
Acceso por bytes

Memoria

No Volátil
Acceso secuencial
Acceso por bloques

Disco (HDD/SSD)

Almacenamiento para BDs

BDs orientadas a disco

Cambios no volátiles

Operar sobre datos: memoria

Trasladar datos desde/a memoria

Leer y escribir de/a disco es costoso

Garantizar desempeño como si la BD viviera en memoria

Memoria

Disco (HDD/SSD)

Discos

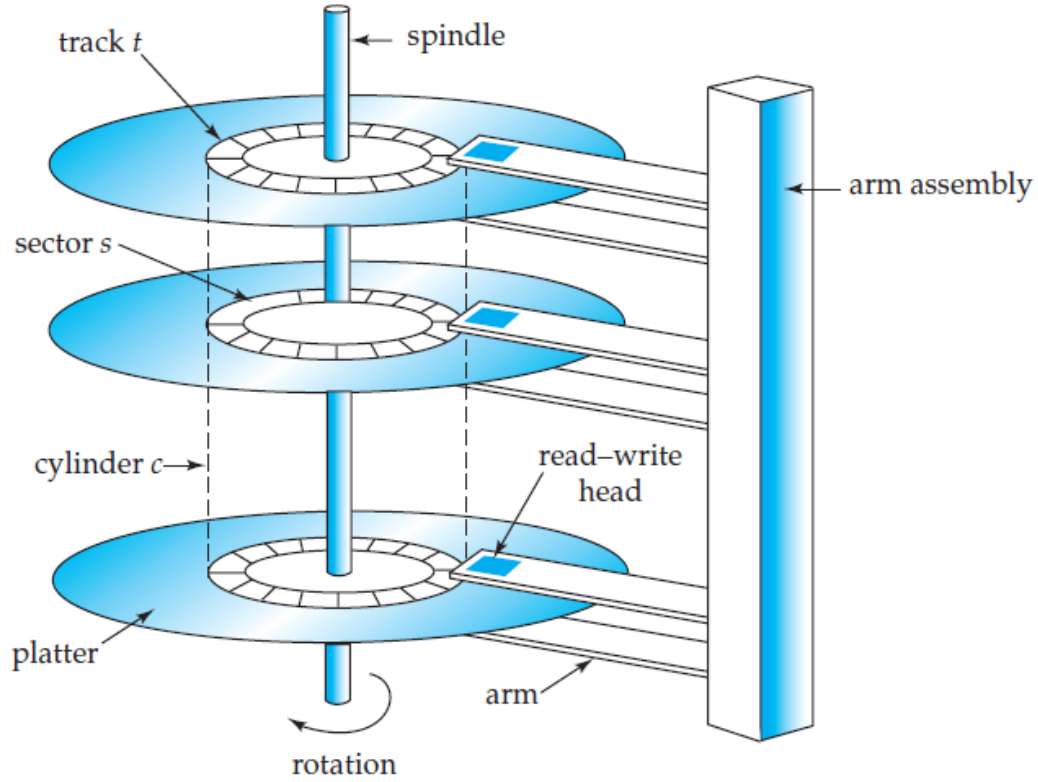
Acceso secuencial en disco es mucho más rápido que acceso aleatorio

Leer por **bloques** de datos en disco, no elementos individuales

Disco ofrece una interfaz (controlador de disco) que le permite recibir instrucciones del computador

HDD: Serial ATA – SATA (estándar)

Disco (HDD)



*Tomado de Databases Systems Concepts

Discos

Disco (HDD)

Tiempo de acceso

Tiempo de búsqueda (seek time): tiempo para reposicionar el brazo

Latencia rotacional: tiempo para girar el disco hasta el sector solicitado (5400, 7200, etc RPM)

Tasas de transferencia

Tiempo de vida medio: mean time to failure (MTTF)

Discos

Acceso aleatorio más rápido que en HDD (Flash tipo NAND)

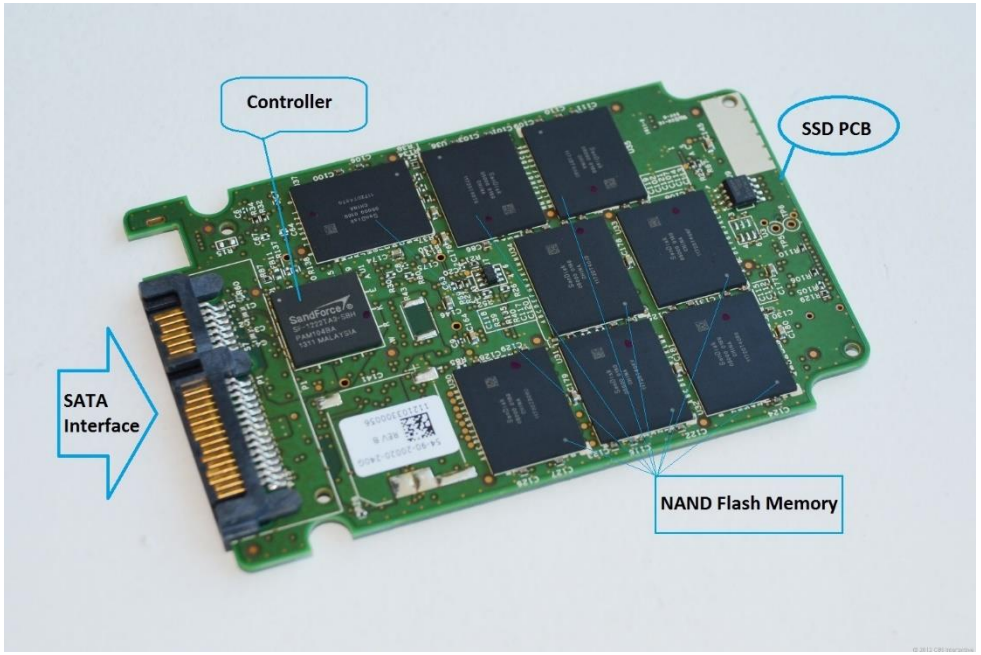
Borrado por zonas (compuestas de muchos bloques) mucho más lento

Cambios: borrado -> re-escritura

Cambios: re-escribe completamente en otra bloque, descarta bloques previos

Borrado masivo programado cuando muchos bloques se han descartado

Disco (SSD)



*Tomado de http://www.angeldatarecovery.com/ssd-recovery/attachment/ssd_diagram/

Discos

Número de veces que se puede borrar un bloque es finito (10E5 – 10E6)

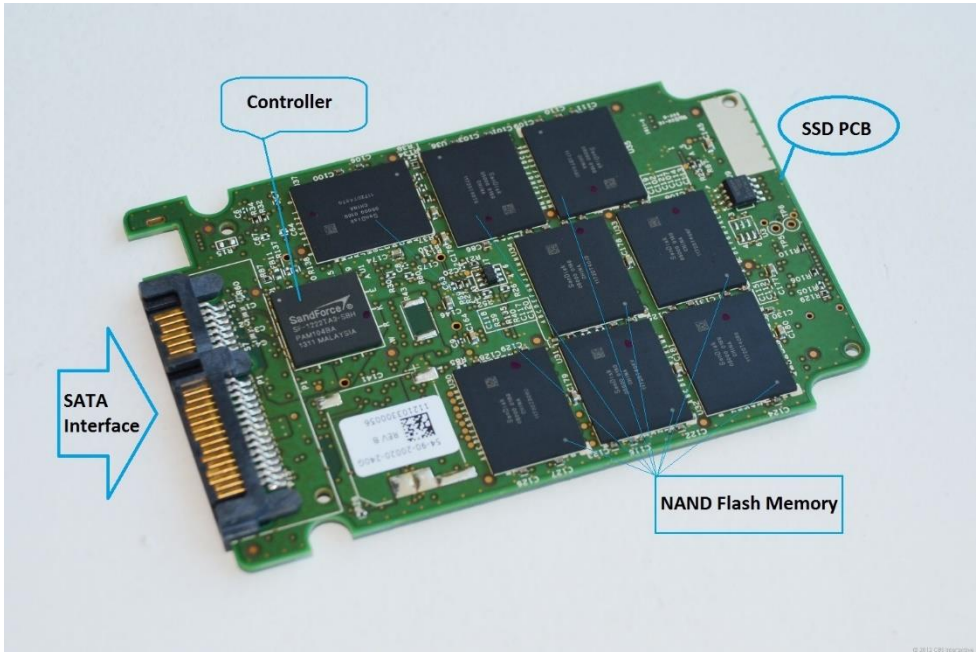
Datos calientes (hot data): en zonas con pocos borrados

Datos fríos (cold data): en zonas con muchos borrados

Nivelado de desgaste

Si una zona no se puede borrar más, se descarta y el controlador la ignora

Disco (SSD)



*Tomado de http://www.angeldatarecovery.com/ssd-recovery/attachment/ssd_diagram/

Confiabilidad

MTTF: mean time to failure – tiempo medio de falla

MTTR: mean time to repair – tiempo medio de reparación

TTF (tiempo de vida): variable aleatoria

Tiempos de vida independientes

Tiempos de vida exponenciales (λ)

Mirrored system (sistema espejo): discos con copias idénticas de la información

Confiabilidad

¿MTTF en un sistema espejo?

¿Distribución del mínimo?

2 discos espejo: exponencial(2λ)

n discos espejo: exponencial($n\lambda$)

MTTF n veces mayor en un sistema con n discos

Reemplazo

Confiabilidad

Hazard function (función de riesgo)

TTF: X

$\lambda(t) = \lim_{dt \to 0} P(t< X < t + \delta t \mid X > t) / \delta t$

Probabilidad de que el disco falle inmediatamente dado que ha sobrevivido hasta este momento t

$\lambda(t) = f(t) / (1-F(t))$

Exponencial: constante

Otras: Creciente/decreciente

Confiabilidad y RAID

RAID: redundant array of independent disks

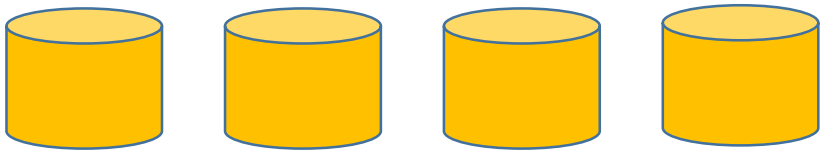
Aprovechar múltiples discos para mejorar desempeño y confiabilidad

Data striping: dividir datos en varios discos para mejorar tasas de acceso

Bit-level data striping: cada byte dividirlo y asignar cada bit a un disco diferente (pueden ser 2/4/8/16/etc discos)

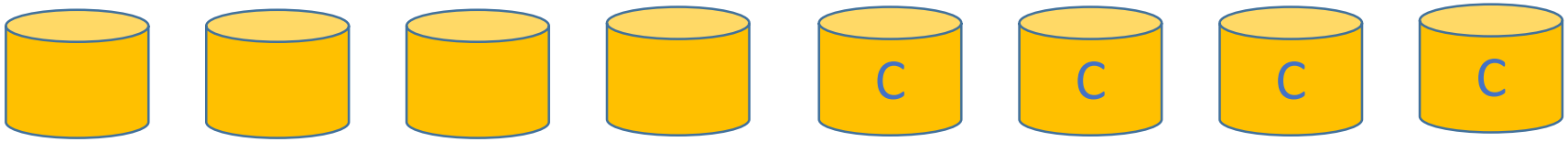
Block-level data striping: asignar bloques de memoria a lo largo de los discos disponibles

RAID 0: striping no redundante

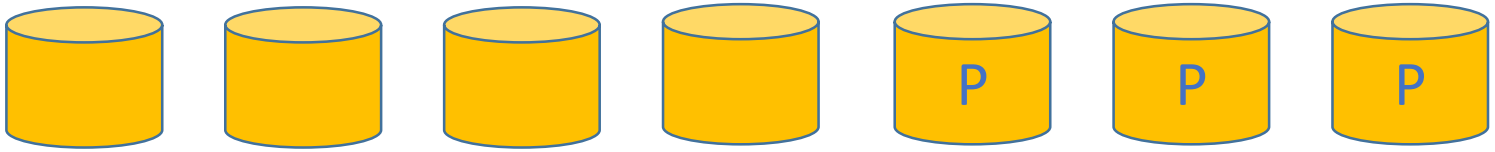


Confiabilidad y RAID

RAID 1: data striping con redundancia (discos espejo)



RAID 2: data striping con paridad (códigos de corrección de error)



- Bits adicionales de paridad y de corrección de errores
- Recuperar información en caso de que se pierda (falla de un disco)

RAID 3 - RAID 4 - RAID 5 - RAID 6

Almacenamiento de Archivos para BDs

DBMS almacena la información como archivos en disco

Archivos binarios en el formato del DBMS

El sistema operativo los trata como cualquier otro archivo

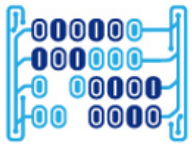
DBMS administrador de almacenamiento

¿Por qué no el SO?

DBMS tiene más contexto sobre la información que se está procesando

Programar hilos y procesos para leer y escribir datos

Soluciones específicas para extraer información con anticipación



Almacenamiento de Archivos para BDs

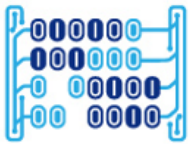
Representar la BD en archivos en disco

DBMS **administrador de almacenamiento AA** (storage manager)

BD guarda la información en un conjunto de **archivos**

Un **archivo** está compuesto por un conjunto de **páginas**

El AA lee y escribe datos en páginas, lleva cuenta de espacio libre, etc



Páginas en BDs

Una **página** contiene un conjunto de **tuplas**

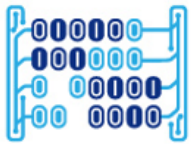
También puede contener otra información: logs, metadatos, etc

Usualmente una página solo contiene un tipo de datos

Cada página tiene un id único (**page id**)

El AA responde solicitudes del tipo **get page # XYZ**

AA mantiene un mapa para saber dónde se almacena cada página (indirection map)



Páginas en BDs

Una **página** contiene un conjunto de **tuplas**

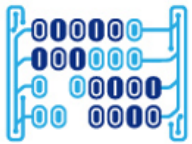
También puede contener otra información: logs, metadatos, etc

Usualmente una página solo contiene un tipo de datos

Cada página tiene un id único (**page id**)

El AA responde solicitudes del tipo
get page # XYZ
write data (in a page)

AA mantiene un mapa para saber dónde se almacena cada página (indirection map)



Páginas en BDs

Cada página tiene un **tamaño fijo** (selección rápida)

Páginas (bloques) a varios niveles (failsafe):

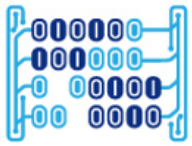
- Hardware (4KB)
- Sistema operativo (4KB)
- DB (1-16KB)

SQLite: 1KB

DB2: 4KB

PostgreSQL/SQLServer: 8KB

MySQL (sin compresión): 16 KB



Páginas en BDs

¿Cómo organizar las páginas?

No ordenado:

- Heap
- Tabla de hashing

Ordenado:

- Secuencial

Archivo Heap para BDs

Colección no ordenada de páginas (tuplas en order aleatorio)

Get página

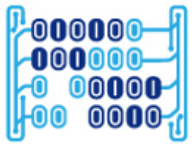
Delete página

Iterar sobre todas las páginas

¿Qué páginas existen? ¿Espacio libre en cada una?

Implementación:

- Lista enlazada
- Directorio de páginas



Archivo Heap para BDs

Implementación como **lista (doblemente) enlazada**

Página encabezado (header page)

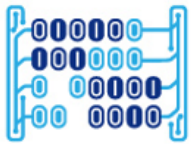
- Apuntador a lista de páginas libres (con espacio)
- Apuntador a lista de páginas de datos (sin espacio)

Get página: ir a lista de páginas de datos

Agregar datos: ir a lista de páginas libres

Cada página lleva la cuenta de su espacio libre

Recorrer toda la lista para encontrar o agregar datos



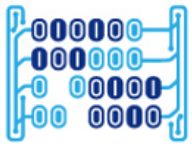
Archivo Heap para BDs

Implementación como **directorio de páginas**

Página especial: directorio con la **ubicación de todas las páginas** (y espacio libre)

Debe mantenerse sincronizada

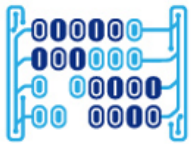
Es posible recuperar la página de directorio en caso de falla



Páginas en BDs

Encabezado: metadatos sobre la página
- Tamaño
- Checksum: revisar integridad de la información
- Tipo/versión de DBMS
- Etc

Datos:
- Tuplas
- Otros (logs,…)



Páginas en BDs

Almacenar tuplas en una página

Tuplas de tamaño fijo vs tuplas de tamaño variable

Almacenamiento por líneas, ¿problemas?

¿Cómo eliminar tuplas?

Páginas en BDs

Páginas por espacios (**slotted pages**)

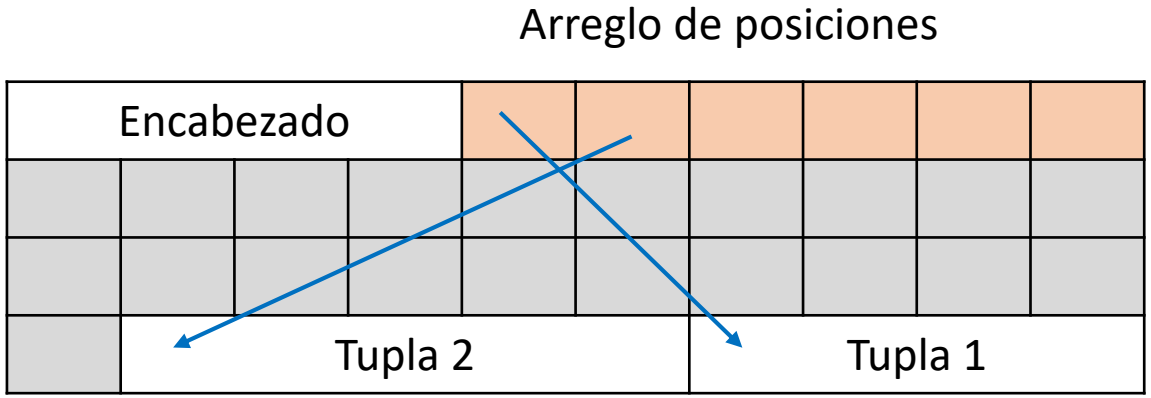
Encabezado al principio (# de espacios usados, ubicación del último espacio lleno)

Luego el arreglo de posiciones (slot array)

Al final de la página se agregan las tuplas

Arreglo de posiciones: mapa

Get Página # + Tupla #

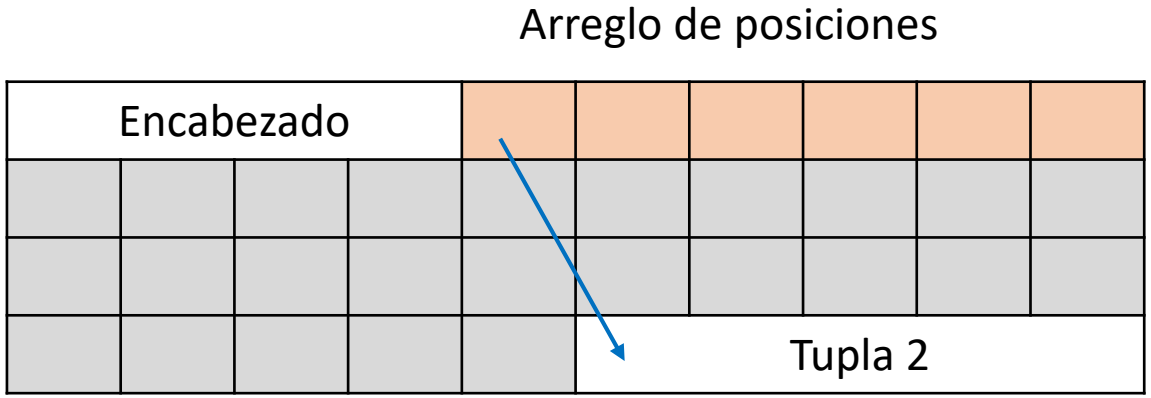


Páginas en BDs

Eliminar tuplas

Agregar tuplas

Leer tuplas



Almacenando Registros

Registros de tamaño variable

Registros de varios tipos

Campos de tamaño variable

Campos repetidos

0	4	8	12	20	21	26	36	45	

Almacenando Registros

Primero va info sobre campos de tamaño variable

Luego campos de tamaño fijo

Luego null bitmap

Luego

Cada registro lleva un par (offset, longitud) para saber dónde va cada registro y de qué longitud es

Longitud en bytes

null bitmap: tantos bits como campos, si uno es null, bit = 1, si no, 0

Almacenando Registros

16,5	21,4	25,4	58	0000	‘12345’	‘Juan’	‘MACC’
0	4	8	12	15	16	21	25 29

estudiante

ID varchar(5)

Nombre varchar(20)

Programa varchar(20)

Creditos numeric(3,0)

Diccionario de Datos

Información acerca de los datos: datos sobre datos: metadata

Diccionario de datos o catálogo del sistema:

- Nombres de las relaciones
- Nombres de los atributos de cada relación
- Dominios y longitudes de cada atributo
- Nombres y definiciones de las vistas
- Restricciones de integridad
- Usuarios de la BDs y sus contraseñas Información sobre Registros de tamaño variable

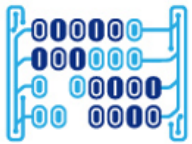
Diccionario de Datos (cont.)

Más metadatos

- Número de tuplas en cada relación
- Método de almacenamiento de cada relación

- Nombre de archivos almacenando cada relación
- ID de página almacenando cada relación

- Esta información es una BDs en sí misma
- Modelo ER de la BD de metadatos



Buffer de la BD

- Minimizar número de transferencias de páginas entre el disco y la memoria
- Mantener muchas páginas en memoria: acceso desde memoria, no desde disco
- Buffer: lugar en memoria donde se almacenan las páginas copiadas desde disco
- Por cada página en memoria, hay una copia en disco, aunque puede ser más vieja
- Administrador buffer (buffer manager)

Administrador del Buffer de la BD

Buffer manager recibe solicitudes por página

Si la página está en memoria, la retorna

Si no:

- El manager reserva espacio en memoria para la nueva página (eliminando un bloque de memoria si hace falta - escribir en disco si es necesario)
- Lee la página de disco a memoria y retorna la dirección en memoria
- Administrador de memoria virtual (SO)
- Pero direcciones en memoria no son suficientes para almacenar todas las páginas de la BDs

Estrategia de reemplazo de Buffer

Seleccionar una página en memoria a eliminar.

LRU (least recently user): eliminar página que no se ha usado hace más tiempo

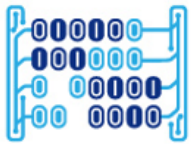
Páginas que se han usado hace poco tienen más probabilidad de ser usadas próximamente

La BDs tiene más información que el SO, puede usar mejores estrategias

Estrategia de reemplazo de Buffer

Ejemplo:

select * from instructor natural join department;

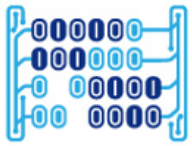


Estrategia de reemplazo de Buffer

```
for each tuple i of instructor do
        for each tuple d of department do
                if i[dept name] = d[dept name] then
                        sea x la tupla a crear
                        x[ID] := i[ID]
                        x[nombre_depto] := i[nombre_depto]
                        x[name] := i[nombre]
                        x[salario] := i[salario]
                        x[edificio] := d[edificio]
                        x[presupuesto_depto] := d[presupuesto]
                        agregar tupla x al resultado
                end-if
        end-for
end-for
```

Estrategia de reemplazo de Buffer

- Instructor y departamento se almacenan en archivos diferentes
- Cada vez que se procese un instructor i, ya no se vuelve a usar
- Cada vez que se termine de procesar una página de instructor, se puede eliminar de memoria
- Estrategia: eliminar inmediatamente (toss-immediate)



Estrategia de reemplazo de Buffer

- Cada vez que se procese un departamento d, solo se vuelve a usar hasta que hasta que se hayan revisado todos los demás departamentos
- Cada vez que se termine de procesar una página de departamento (si se necesita eliminar una) se puede eliminar la más reciente
- Estrategia: most recently used (MRU) -> necesario fijar página actualmente en uso (desfijar una vez se procesan todas las tuplas de departamento en la página)

Estrategia de reemplazo de Buffer

Otras:

- No eliminar de memoria páginas asociadas a metadatos

- No eliminar de memoria páginas asociadas a índices de búsqueda