

## Medidas de Rendimiento de Discos Magneticos

### 1- Tiempo de acceso:

Tiempo que tarda en leer o escribir en datos desde tu petición hasta su confirmación

### 2- Tiempo de Búsqueda:

Tiempo que tarda en ubicar el plato en la pista correcta

### 3- Tiempo de Transferencia:

Volumen de Dato por unidad de tiempo que se puede cargar o recuperar del Disco

### 4- Tiempo Medio de Fallo:

Japón de tiempo que puede trabajar un disco continuamente sin fallar. De 3 a 5 años.

### 5- Latencia Rotacional:

Tiempo que se tarda en aparecer un sector a leer o escribir debajo de los cabezales del plato

## Enfoques de Organización de Registros en Archivos

- \* Heap: los registros son guardados en cualquier lugar en donde existe espacio suficiente.
- \* Secuencial: los registros se guardan en un orden secuencial de acuerdo a una clave de búsqueda
- \* Hash: los registros son almacenados en un bloque especificado por una función Hash
- \* Organización por agrupación: los datos de diferentes relaciones se guardan en el mismo archivo

# Redundancia de Discos

Raid: Redundant Array of Independent Disk

Provee una visión con:

\* Alta Capacidad \* Alta Velocidad \* Alta Confiabilidad

Raid 1: el esquema de Almacenamiento basada en RAID 1

Disposición espejada, cada disco tiene su espejo

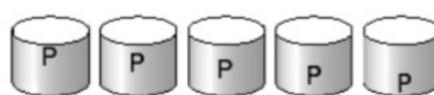
- Tiene el mejor rendimiento de escritura que el Nivel 5
- Elección para aplicaciones que requieren clústeres de almacenamiento y grandes volúmenes de escritura y lectura
- Ideal para casos en donde la seguridad de los datos es más importante que el desempeño de las escrituras.



(b) RAID 1: Discos con imagen

Raid 5: Paridad distribuida con bloques entrelazados entre n

- Divide la paridad y los datos en los  $N+1$  discos.
- Todos los discos pueden atender las solicitudes de lectura.
- En cada conjunto de  $N$  Bloques lógicos, uno de los discos guarda la paridad y los otros  $N-1$  guardan los bloques.



(f) RAID 5: Paridad distribuida con bloques entrelazados

## Organización dentro del Archivo

Registros de longitud Fija: Cada registro empieza en  $n*i$  bloque donde  $n$  es la longitud de los registros e  $i$  el número del Registro. Los registros se agregan siempre al final del archivo.

Para eliminación:  
\* Mover los registros  $i+1$  hasta  $n$ , una posición atrás  
\* Mover el registro  $n$  a la posición  $i$

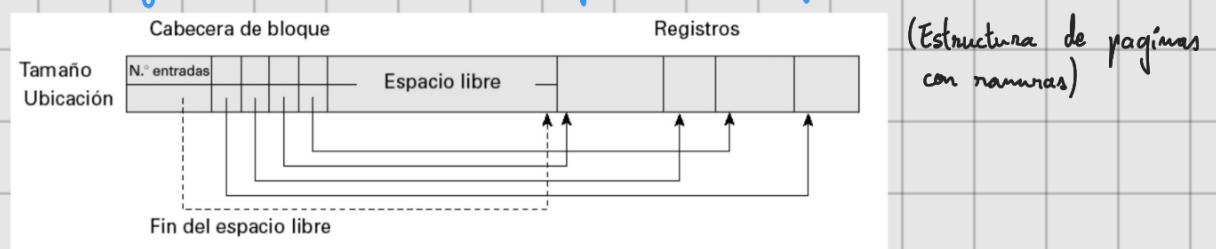
\* Dejar espacio libres pero manteniéndolos enlazados con una lista libre

Lista libre: \* Se utiliza el primer registro del bloque como cabecera del mismo  
Allí se guarda el primer registro libre, entre otros datos

\* Cada registro libre tiene un puntero que apunta al siguiente libre

Registros de longitud Variable: No utilizados cuando se quieren guardar múltiples tipos de registros en una tabla

Cuando los registros contienen campos de diferentes tamaños



## Tipos de Índices.

Índice Ordenado: Los registros índice están ordenados al valor de la clave de búsqueda. Puede ser Primario o secundario

Índice primario: Ordena físicamente la tabla

Índice Secundario: NO ordena físicamente la tabla

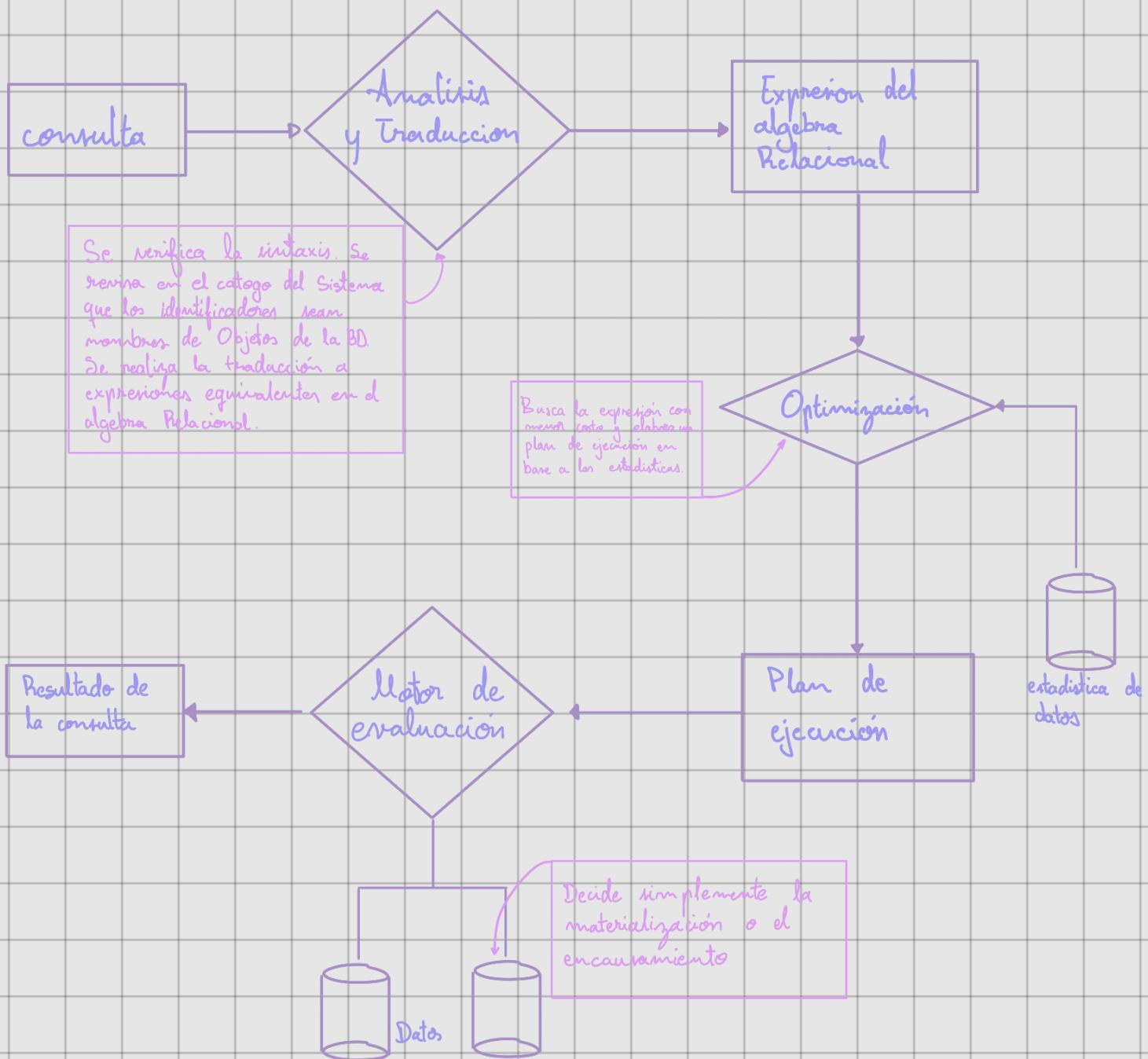
Conviene usarlos para Búsquedas en Rango

Índice Asociativo o Hash: Los registros índices están almacenados en Bloques de acuerdo a una función Hash  
Útiles en condiciones de igualdad.

Índice Bitmap: Son un tipo especial de índices diseñados para consultar eficientemente en múltiples claves. En su forma más simple un índice bitmap tiene un array de bits para cada valor del atributo índice.

Son útiles tanto para consultar sobre un atributo o sobre varios, siempre y cuando haya un índice bitmap en cada atributo.

# Pasos para Procesamiento de una Consulta.



\* Número de tuplas de la relación  
Nr

\* Número de bloques que contienen tuplas de la relación r.  
br

\* Tamaño en byte de una tupla r.  
tr

\* Número de tuplas de r que caben en un bloque.  
fr

\* Nro de valores distintos del Atributo A de la relación r  
V(A,r)

\* Nro media de filas en la tabla  
 $CS(t,r) = Nr / V(A,r)$

\* Grado de salida de los nodos internos del índice i  
gi

(Para índices con estructura  
= ran de Árbol B/B+)

\* Nro bloques en el Nivel hoja  
MBi

\* Altura del índice

$$AA_i = \lceil \log_{gi} (V(A,r)) \rceil$$

## Algoritmos de Ordenación

A1. Busqueda lineal: Cuando no se tiene un índice de clave de Busqueda ni se tiene ordenado la tabla. Necesariamente se revisan todos los Bloques.

(Full Scan)

Costo =  $b_r$  (cantidad de Bloques del Registro)

A2. Busqueda Binaria: Cuando no se tiene un índice de clave pero la tabla esta ordenada segun la clave de Busqueda.

$$\text{costo} = \underbrace{\lceil \log_2 b_r \rceil}_{\text{localizar el Primer Bloque}} + \underbrace{\lceil CS(A, r) / f_r \rceil - 1}_{\substack{\text{Repetición Promedio del dato}}}$$

localizar el Primer Bloque

Nro de Bloques subsecuentes ya que la clave es no candidata y ordena la tabla

A3. Busqueda en Índice primario para un atributo clave: cuando se tiene un índice de clave de Busqueda que ademas es un atributo clave.

$$\text{costo} = \underbrace{AA_1 + 1}_{\substack{\text{recorren el arbol}}} \rightarrow \text{lectura del Bloque}$$

A4. Busqueda en Índice secundario para atributo no Clave: Cuando no se tiene ningun índice de clave de Busqueda ni tampoco un atributo clave.

$$\text{costo} = \underbrace{b_r}_{\substack{\text{bloque}}} + \underbrace{CS(A, r)}_{\substack{\text{Nro de filas}}} + 1$$

# Algoritmos de Join

## Algoritmos de Join

(Join anidado)

Tuplas

Bloques

LRU

MRU

- Nested Loop: costo:  $b_r + n_r * b_s$       costo:  $b_r + b_r * b_s$

$$b_r + b_r * b_s$$

$$b_r + n * (b_s - M + 1)$$

n: numero de Bloques

M: Memoria disponibles

- Index Join: costo:  $b_r + n_r * [\log_{P/2}(n_s)] + 1$       P: Punteros

- Range Join: costo:  $b_r + b_s + b_r * (2 * [\log_{M-1} b_s] + 1)$  cant. de Memoria

- Hash Join: costo:  $3(B_r + B_s) + 2 \frac{B_s}{M} 1,2$ ; precondición de que el particionamiento entra en memoria  
 $= (B_r + B_s) \cdot (2([\log_{M-1} (B_s)] - 1) + 1)$  cuando no entra en memoria

$$\text{tuplas} = k$$

$$\frac{\text{tuplas}}{\text{capacidad}} \rightarrow \frac{\text{total de Tuplas}}{\text{tuplas P/Bloque}} = \text{total Bloque}$$

# Algebra Relacional

## Selección ( $\sigma$ )

$\sigma_{[\text{condición}]} (\text{Tabla})$ : ejemplo:  $\sigma_{\text{saldo} > 1000} (\text{Caja_de_ahorro})$

## Proyección ( $\Pi$ )

$\Pi_{[\text{columnas}]} (\text{Tabla})$ :  $\Pi_{\text{Nombre, cuenta}} (\text{Tabla})$

## Join ( $\bowtie$ )

$[\text{tabla}] \bowtie_{[\text{condición}]} [\text{tabla}]$ : Alumno  $\bowtie_{\text{a.ci}=\text{e.ci}}$  Examen

Todas las operaciones generan tablas

Cuando no se especifica la condición es full Join.