Tema 6: Organización de Ficheros: Organizaciones Base

- **Introducción. Objetivos y Parámetros.**
- **Organizaciones básicas (de naturaleza consecutiva)**
  - Organización sin orden: Org. **Serial**
  - Organización ordenada: Org. **Secuencial**
- **Organizaciones Direccionadas**
- **Agrupamiento físico: organización en *Cluster***
- **Procesos sobre Organizaciones Base**

Tema 6.1: Introducción

Queremos Optimizar…

- ***Tiempo de respuesta***: *(en Actualización / en Recuperación)*
  - ***disminuir los accesos*** *a soporte (organizaciones, memorias intermedias…)*
  - otros recursos (CPU; RAM): procesos costosos, ordenaciones, ensamblajes …
- *Espacio de Almacenamiento*
  - Incrementar densidad, minimizar almacenamiento auxiliar, …
  - ¡El espacio de almacenamiento influye en el Tiempo de Respuesta!
- *Coste de Desarrollo y Mantenimiento*

Partimos de ciertos requisitos…

- Características del dispositivo: *bloque, tiempo acceso, (operaciones),…*
- Características de los archivos: *tamaño registro, cardinalidad, volatilidad,…*
- Características de los procesos: *tipología, frecuencia, criticidad, …*

Tema 6.2: Organizaciones Consecutivas

Organización básica: Organización Serial

Surge con los dispositivos de almacenamiento seriales:

Dispositivo Serial: proporciona registros físicos en serie, esto es, que se registran uno detrás de otro, y se acceden en ese orden

Instrucciones: leer (bloque) y reset

***Ejemplo** de soporte serial*: la cinta magnética

Organización Serial: almacenar registros sin criterio de colocación

Aprovechamiento de Espacio: **ÓPTIMO**

→ Coste de accesos a la totalidad: **ÓPTIMO**

No existen claves privilegiadas → ***no se puede filtrar*** (cjto. dir. relevantes)

Tamaño área de búsqueda: **n** bloques (consecutivo) | **N** cubos (no-consecutivo)

Tema 6.2.1: Procesos Org. Serial

Actualización:

- inserción: se añaden registros al final del fichero
- borrado:
  - borrado físico: se *vacía* el registro → se desplaza el resto
  - borrado lógico: se *marca* el registro → se genera un *hueco*
- modificación:
  - registros fijos: se altera el contenido
  - serial no consecutiva: si hay hueco, se modifica en el mismo cubo
  - otros casos: se borra el antiguo y se reinserta modificado

Recuperación:

- consulta selectiva identificativa: leer todo, hasta encajar el primer registro
- consulta selectiva no identificativa:
- consulta selectiva multiclave:
- consulta a la totalidad:

leer todo el fichero

Tema 6.2.1: COSTES Org. Serial

Actualización:

- inserción: **1** acceso
- borrado (de *k* registros): ***selección*** + ***k*** accesos
- modificación (de *k* registros):
  - registros fijos o serial no consecutiva (con eld): ***selección*** + ***k*** accesos
  - otros casos: **(*selección* + *k* + *k*)** accesos

Recuperación: el coste de la selección (***selección*** accesos)

Coste de Seleccionar Registros: ¡sin clave privilegiada no hay filtrado!

- selección identificativa: **(N+1)/2 accesos** (no consecutiva); (n+1)/2 (consec.)
- consulta selectiva no identificativa:
- consulta selectiva multiclave:
- consulta a la totalidad:

} **N accesos** (n si es consecutiva)

Tema 6.2.1: Mantenimiento Serial

Gestión de Huecos

Espacio Libre Distribuido:

- Mover registros (al modificarlo) es engorroso (y genera costes adicionales)
- En orgs. no consecutivas, se puede evitar dejando ELD para modificaciones:
  - *porcentaje de espacio que sólo puede ser utilizado en op. de modificación*
- En el DBMS Oracle, se denomina PCTFREE, y viene por defecto al 10%

Gestión de Huecos (gap-list / *pila de inserción*):

- Estructura en memoria principal que mantiene localizados los *huecos*
- En Oracle, un cubo tiene hueco si su densidad baja de un umbral (PCTUSED)
- Insertar tendrá mayor coste (~2 accesos)… pero reduce el tamaño del fichero
- La lista de huecos se construye en el primer *full scan* (o por sondeo de cubos)

Compactación:

- Proceso de mantenimiento que desplaza registros para eliminar huecos y restablecer el ELD si fuera necesario

Tema 6.2.2: Organización ordenada

Organización Secuencial

Surge a partir de la serial introduciendo un orden (clave de ordenación física).
Se precisa un dispositivo capaz de acceder aleatoriamente a los bloques
(bien de acceso secuencial, con op. de avance y retroceso, o bien de acceso directo).

- El acceso aleatorio a bloques obliga a contar con un mecanismo para interpretar su contenido (localizar el comienzo del primer registro)
  - a nivel físico (por *cubos*): el bloque comienza con un reg. completo
  - a nivel físico-lógico (registros consecutivos): marca de inicio/fin

Organización Secuencial: almacenar registros con criterio de orden

Aprovechamiento de Espacio: **ÓPTIMO** (con excepciones)

→ Coste de accesos a la totalidad: **ÓPTIMO**

Clave privilegiada de orden → ***búsqueda dicotómica***

Tamaño área de búsqueda: **n** bloques (consecutivo) | **N** cubos (no-consecutivo)

Tema 6.2.2: Procesos Org. Secuencial

Actualización:

- inserción no ordenada: se añaden registros al final del fichero.
  La org. **degenera**: *área desordenada*, reduce progresivamente la eficiencia.
- inserción ordenada: se localiza la ubicación del registro (busq. dicotómica). Si el registro no cabe, desborda (requiere gestión de desbordamientos).
- borrado: como en organización serial (excepto la selección)
- modificación: como en org. serial con una excepción:

  – *modificar la clave de ordenación*: borra reg. antiguo + reinserta modificado

Selección:

- consulta por clave no privilegiada: como en la org. serial
- consulta por clave privilegiada (ordenación)
  - clave identificativa: búsqueda dicotómica
  - clave no identificativa: búsqueda dicotómica extendida
- consulta selectiva multiclave: primero filtra, luego aplica b. dicotómica
- consulta a la totalidad ordenada (por clave privilegiada): óptima

Tema 6.2.2: Búsqueda Dicotómica

Búsqueda Dicotómica:

- mirar el elemento (bloque/cubo/registro) central en el espacio de búsqueda
- si coincide→ fin; si no, restringir espacio de búsqueda a la mitad relevante
- volver a empezar (sobre la mitad escogida)

Búsqueda Dicotómica Extendida:

- Buscar primer elemento (por *búsqueda dicotómica*)
- Buscar serialmente hacia adelante hasta encontrar uno distinto (*fallo*)
- Buscar serialmente hacia atrás hasta encontrar uno distinto (*fallo*)

Tema 6.2.2: Coste Busq. Dicotómica

Número de accesos con la Búsqueda Dicotómica:

- *consideramos el peor caso:* $n^o accesos_{max} = \lceil \log_2 (x+1) \rceil$

- El número de elementos en la búsqueda (x) depende de la relación físico-lógica:

	Consecutivo		No Consecutivo	
k: registros/valor	$f_b \geq k$	$f_b < k$	$T_c \geq k$	$T_c < k$
CO identificativa (k=1)	# bloques	# registros	# cubos	
CO no identificativa	# bloques	# valores (CO)	# cubos	# valores (CO)

- En resumen, x = MIN (#valores(CO), N)

Número de accesos con la Búsqueda Dicotómica Extendida:

- *fichero No Consecutivo:* $n^o accesos_{max} = \lceil \log_2 (x+1) \rceil + \lceil \frac{(k+1)}{T_c} \rceil$

- *fich. Consecutivo:* $n^o accesos_{max} = \lceil \log_2(x+1) \rceil + \lceil \frac{T_{reg} \cdot (k+2) - 1}{T_{bq}} \rceil \approx \lceil \log_2(x+1) \rceil + \lceil \frac{k+2}{f_b} \rceil$

Tema 6.2.2: COSTES Secuencial

Actualización:

- inserción:
  - ordenada: $log_2 (x+1) + 1$ [+ coste desbordamiento]
  - no ordenada: **1** acceso (inserción serial en área desordenada)
- borrado (de *k* registros): ***selección*** + $\boldsymbol{k/T_c}$ accesos
- modificación (de *k* registros):
  - no consecutiva (con eld) y no modifica CO: ***selección*** + $\boldsymbol{k/T_c}$ accesos
  - otros casos: coste borrado + coste inserción

Recuperación: el coste de la selección (***selección*** accesos)

Coste de Seleccionar Registros (por clave privilegiada)**:**

- selección identificativa: $log_2 (x+1)$ [+ coste desbordamiento]
- consulta selectiva no identificativa (CO no identif. o consulta en un rango): *coste de la busq. dicotómica extendida* [+ coste desbordamiento]
- consulta selectiva multiclave: primero filtra (reduce *x*) y luego búsq. dicótomica
- consulta a la totalidad ordenada (por la CO): coste serial (full scan, N)

Tema 6.2.2: Mantenimiento Secuencial

Gestión de desbordamientos:

- Org. Consecutivas: área de desbordamiento (área desordenada)
- Org. No Consecutivas:
  - *rotaciones*: traspasar elementos de un cubo lleno a su vecino (si tiene espacio libre…)
  - intercalar cubos completamente vacíos (al crear o reorganizar el fichero)
  - *partición celular*: cuando desborda, se intercala un cubo vacío y se reparten los regs.

Espacio Libre Distribuido para inserción:

- En org. secuenciales, reduce la tasa de desbordamiento
  - *porcentaje de espacio libre en reorganizaciones y cargas masivas*

Lista de Huecos:

- Los huecos tienen orden, tamaño y localización → poco reutilizables
- En org. no consec., esta lista es un índice no denso que indica el espacio libre

Reorganización:

- Compacta y reescribe todos los registros ordenados (mezclando ambas áreas)
- Coste elevado; habitualmente se aplican alg. basados en alm. auxiliar

Tema 6.3: Organizaciones Direccionadas

Surge con los dispositivos de acceso aleatorio (directo):

Acceso aleatorio: proporciona el registro físico indicado.
Algunos dispositivos (disco duro) mantienen ventajas seriales.

Instrucciones: leer(x) / escribir (x), donde x es la dirección física del bloque

Ejemplos: *disco duro (tambor), SSD (SLC/MLC), …*

Organización direccionada: ubicar cada registro en 'su sitio'

Clave privilegiada, la CD → gran capacidad de ***filtrado*** (cjto dir relevantes)

Aprovechamiento de Espacio: **reducido**

→ Coste de accesos a la totalidad: **elevado**

Espacio de Direccionamiento: **N** cubos **(siempre es no-consecutivo)**

Ejemplo: el registro cuyo DNI es 1234567 8 se almacena en el cubo #12345678

Tema 6.3.1: Tipos de Direccionamiento

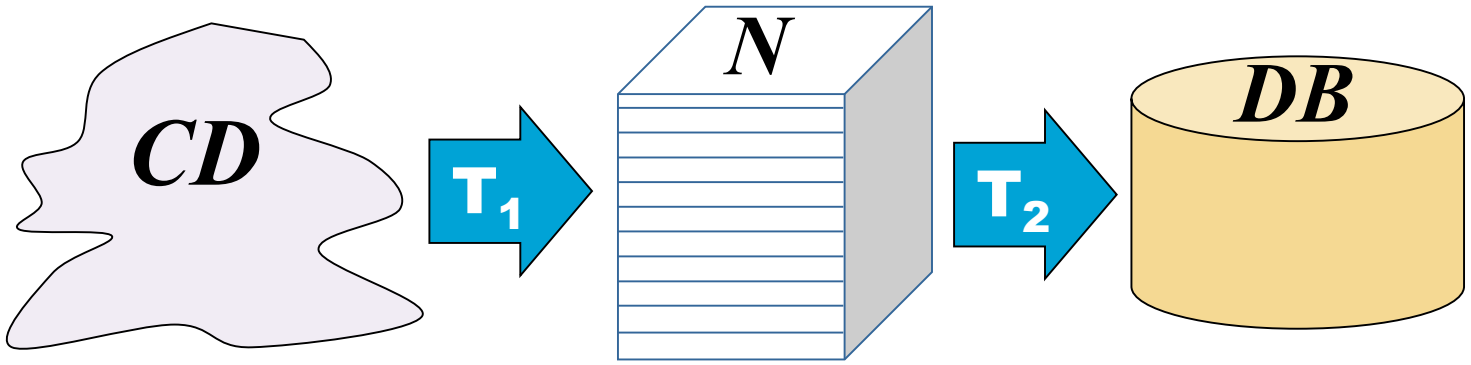
1. *Organización Direccionada Directa:*

cada registro tiene su dirección reservada

- Cada cubo contiene un solo registro
  - Si el tamaño máximo del registro es mucho menor que el del bloque, se pueden considerar **celdas** (*cubos* que son una división del bloque)
- Los valores de CD que no ocurren implican cubos vacíos (baja densidad)
  - Si sólo ocurre un rango de valores, se puede **transformar la dirección**:

Org. Direccionada Directa Absoluta: la CD es la dirección del cubo

Org. Direccionada Directa Relativa: existe una biyección entre CD y dir. cubo



Tema 6.3.1: Tipos de Direccionamiento

2. *Organización Direccionada <u>Dispersa</u> (HASH):*

*La CD **se transforma** en dir_cubo, pero la función **no es una biyección***

- Si la distribución es buena, aumentará la densidad (*dens. ocupación*)
- El *Algoritmo de Transformación* T_1: consta de dos pasos:
  a) conversión numérica: por código ASCII (por ejemplo)
  b) función de dispersión (f_t): proporciona un número de 0 a N-1
     - El objetivo es acercarse a la dispersión uniforme (ideal).
     - Lo opuesto a la dispersión uniforme es producir *cúmulos* y *cubos vacíos*
     - Funciones tradicionales: residuo, truncamiento, plegado,…
     - Métodos de organización 'automática': Lin

- Si la densidad es baja, se puede reorganizar cambiando:
  - la dispersión (función de transformación o T_1 completo)
  - el diseño del cubo: menor E_c → mayor densidad, y más desbordamientos…
  - el espacio direccionamiento: menor N → mayor densidad, y más desbord…

Tema 6.3.1: Tipos de Direccionamiento

- ***Ejemplos:***
  - *Dir. Directo absoluto con CD:NIA.*
    No se transforma, luego N=10^9 **(demasiados cubos)**
  - *Dir. Directo relativo con CD:NIA y f_t: truncamiento a 6 últimos dígitos.*
    N=10^6, pero: cubos vacíos + un registro por cubo = ***baja densidad***
  - *Dir. Disperso con CD:NIA y f_t: truncamiento a 2 últimos dígitos.*
    N=100, pero producirá muchos ***desbordamientos***

- ***Conceptos:***

– Claves sinónimas (para cierta f_t): las que producen la misma dirección

– Claves homónimas: tienen el mismo valor (siempre van a la misma dir.)

– Potencia de direccionamiento: **#valores(CD) ≥ N**

– Colisión: inserción de un registro en un cubo no vacío

– Desbordamiento: colisión en un cubo sin espacio suficiente para insertar

Tema 6.3.2: Procs. Direccionamiento

Actualización:

- inserción: se calcula la dirección, y se añade el registro allí; Si no cabe, **desborda**
- borrado: se localiza el registro y se elimina (aumenta el espacio libre en su cubo).
- modificación: se localiza el registro y se modifica (si no cabe, borrado+reinserción)

Recuperación:

- Localización: filtra por CD; dependiente de ***política desbordamiento***
  - consulta selectiva identificativa: leer cubos no filtrados, hasta encaje: (x+1)/2
  - consulta selectiva no identificativa: leer todos los cubos no filtrados
  - consulta selectiva multiclave: filtrado multiclave
  - otras consultas: *full scan*

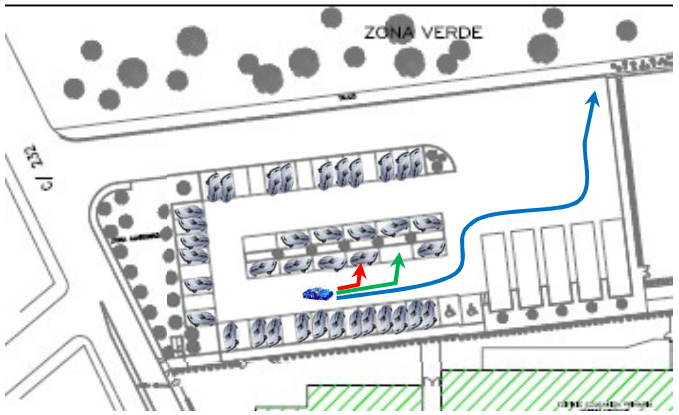
Tema 6.3.2: Gestión Desbordamiento

Las *Políticas de Gestión de Desbordamiento* se pueden clasificar por dos criterios:

1. **Según la zona donde se ubique el registro desbordado**
   - **a)** Saturación: otra dirección dentro del espacio de direccionamiento
   - **b)** Área Desbordamiento: fuera del área de datos (en otro archivo)

Ejemplo:
mi plaza de aparcamiento es *cód_empleado* DIV 2
(la misma plaza se asigna a dos empleados).
Si cuando llego está ocupada, para aparcar puedo:

a) Aparcar en la siguiente plaza
b) Aparcar en unas tierras ahí al lado



2. **Según el mecanismo de ubicación:**
   1. Direccionamiento abierto: *dir_nueva = dir_vieja + 1*
   2. Encadenamiento 

   3. Otros (organización independiente) 

Tema 6.3.2: Gestión Desbordamiento

	Saturación	Área desb.
Dir abierto	✓	✗
Encaden.	✓	✓
Otros	✗	✓

Saturación con Direccionamiento Abierto:

- La nueva dirección se averigua a partir de la dir. desbordada.
  - Sondeo lineal: *dir' = dir + 1*
  - Rehashing: *dir' = dir + k*
  - *Doble hash: dir' = f_{t2}(CD)*
- Si esta estuviera ocupada se produce un ***choque***. Si además no cabe, será un ***rebote.***
- Si se produce un rebote se buscará otra ***dirección nueva*** hasta encontrar una posición libre o hasta haber recorrido todo el espacio (área de datos saturada). Si el área de datos está saturada se precisa otra gestión de desbordamientos.
- Una búsqueda podría recorrer todo el área de datos.
  - Esta técnica no es compatible con el filtrado.
  - En vez de filtrar, se reorganiza el cjto. resultado para ser recorrido en otro orden.
  - Byte de desbordamiento: en cada cubo, indica si ese cubo ha desbordado.
    Toda búsqueda termina al llegar a un cubo no desbordado.
    En media, se recorrerán: N / (#cubos$_{(B_desb=0)}$ + 1) → la organización degenera

Tema 6.3.2: Gestión Desbordamiento

	Saturación	Área desb.
Dir abierto	✓	✗
Encaden.	✓	✓
Otros	✗	✓

Saturación Progresiva Encadenada:

- La nueva dirección se deja apuntada en el cubo desbordado.

- Puntero: información que indica la ubicación de otra información (registro)
  - Puede ser lógico (clave identificativa), relativo (*dir* ∈ [0..N-1]), o físico
  - El puntero relativo de precisión simple indica **en qué cubo** está almacenado
  - El ptro. relativo de precisión doble indica **en qué cubo y en qué posición**

- La saturación encadenada apunta individualmente a registros desbordados (encadenamiento a registro), luego requiere punteros relativos de precisión doble.
- El cubo desbordado tiene un solo puntero. Cuando desborda un segundo registro, éste apuntará al primer desbordado, y el cubo apuntará al nuevo desbordamiento (las inserciones en la cadena de desbordamiento se hacen por la cabeza, no por el final)
- El cubo desbordado puede tener varios punteros (*lista de encadenamiento*) pero no muchos…

Tema 6.3.2: Gestión Desbordamiento

	Saturación	Área desb.
Dir abierto	✓	✗
Encaden.	✓	✓
Otros	✗	✓

Área de Desbordamiento Independiente:

Los registros desbordados son almacenados fuera del área de datos, en un archivo aparte (área de desbordamiento).

Ventaja: se eliminarán los choques (y los rebotes)

Desventaja: es preciso usar más espacio (**auxiliar**).

La organización degenera, y baja la eficiencia en la localización:

- en búsquedas por clave privilegiada, esta área no se filtra
- En búsquedas por clave alternativa, se tienen más cubos

- Generalmente, la organización del área de desbordamiento es serial.
  Pero también podría considerarse una clave privilegiada (CO/CD) con los mismos atributos (o un subconjunto) de la clave privilegiada principal.
  - si la org. secundaria desborda, se requiere otra gestión de desbordamientos, o desencadenar una reorganización automática de todo el fichero.

Tema 6.3.2: Gestión Desbordamiento

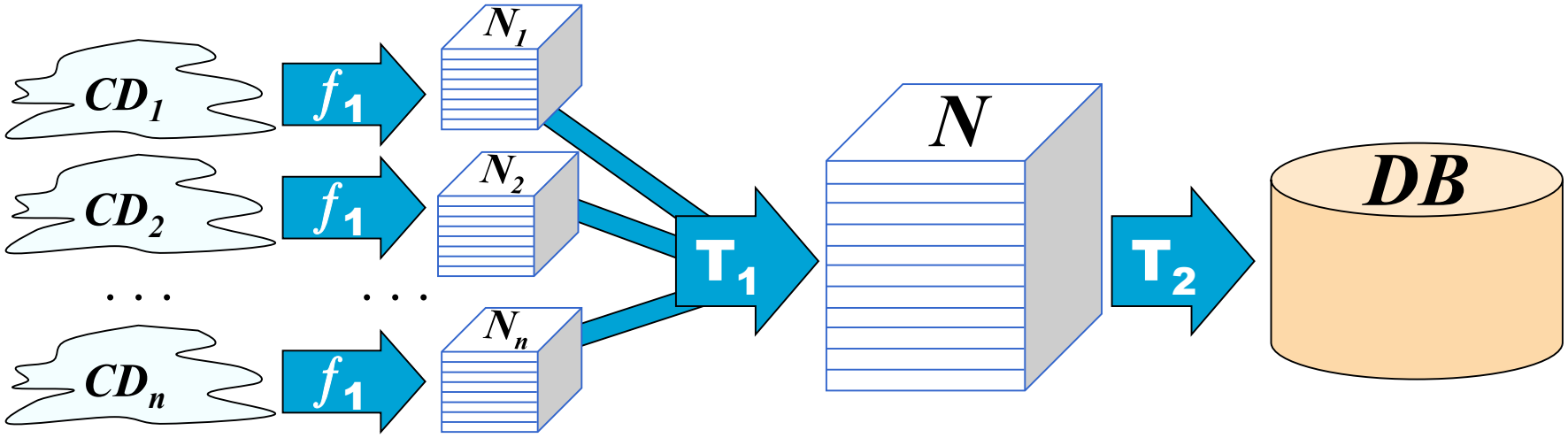
	Saturación	Área desb.
Dir abierto	✓	✗
Encaden.	✓	✓
Otros	✗	✓

Encadenamiento en Área de Desbordamiento:

- Los registros que desbordan se almacenan en un área aparte, y su dirección se deja apuntada en el cubo desbordado.
- Encadenamiento **a registro**: los registros en área de desbordamiento se almacenan serialmente, pero incorporan un puntero de encadenamiento
- Encadenamiento **a cubo**: cuando un cubo desborda, se le asigna a esa dirección un cubo completo dentro del área de desbordamiento:
  - El puntero de encadenamiento es de precisión simple, y se almacena en el cubo
  - El cubo encadenado sólo contiene registros de la dirección que lo apunta
    - ➢ menor densidad (esto favorece a procesos por CD, pero perjudica al resto)
  - También se denomina *extensión del cubo de datos*
  - Un cubo grande (Ec elevado) aprovecha la secuencialidad del disco duro, pero extender cubos pequeños se adapta mejor al contenido de cada cubo (b. equilibrio)

Tema 6.3.3: Dir. Disperso Multi-Clave

- Consiste en ampliar el algoritmo de transformación, para operar varias CD.
- T_1 debe combinar las dispersiones de varias CD_i sobre otros tantos subespacios de direccionamiento (N_i) por otras tantas funciones de dispersión (f_i).
- La combinación suele ser simple: $T_1 \equiv \sum_{i=1}^{n} \left(f_i(CD_i) \cdot \prod_{j=1}^{i-1} N_i \right)$
- El *espacio de direccionamiento global* **N** será la multiplicatoria $N \equiv \prod_{i=1}^{n} N_i$



Tema 6.3.3: Filtrado en Disp. Multiclave

Ejemplo de inserción:

Sean CD_1: nombre, CD_2: apellido1, y CD_3: apellido2,
con $N_1 = 2^2$, $N_2 = 2^3$, y $N_3 = 2^1$
(en total 6 bits de dirección)

d_5	d_4	d_3	d_2	d_1	d_0
apellido2		*apellido1*		*nombre*	

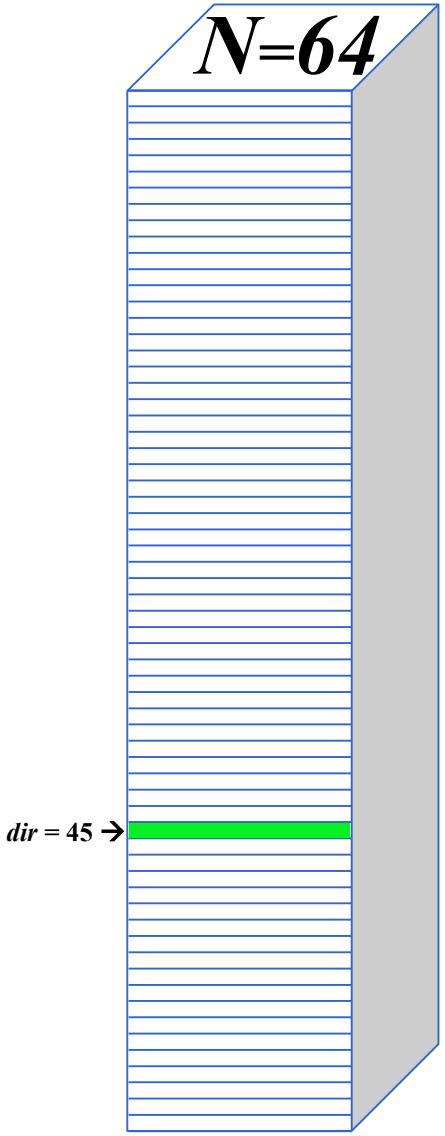
f_1 (‘*John*’) = 1 | (0 1)
f_2 (‘*Pérez*’) = 3 | (0 1 1)
f_3 (‘*Smith*’) = 1 | (1)

$dir = 1 \cdot 2^2 \cdot 2^3 + 3 \cdot 2^2 + 1 = 45$

1	0	1	1	0	1
apellido2		*apellido1*		*nombre*	

Ejemplos de filtrado:

Buscamos a John Pérez Smith (almacenado en el cubo 45)
Cuando buscamos…



Tema 6.3.3: Filtrado en Disp. Multiclave

- Observar que si CD: Nombre+Apellido (direccionamiento simple) sólo podemos filtrar cuando disponemos de ambos. Pero si CD_1: Nombre y CD_2: Apellido, podemos filtrar sólo con conocer uno de esos atributos.
- El algoritmo de filtrado se basa en un vector de booleanos (N posiciones) inicializado a false (0); para *d* claves tendrá otros tantos bucles anidados:

```
∀ k, CR[k]:=0;
If CD1<>"" → {n1_inf := f1(CD1); n1_sup := f1(CD1)}
             ELSE {n1_inf := 0; n1_sup := N1-1};
If CD2<>"" → {n2_inf := f2(CD2); n2_sup := f2(CD2)}
             ELSE {n2_inf := 0; n2_sup := N2-1};
If CD3<>"" → {n3_inf := f3(CD3); n3_sup := f3(CD3)}
             ELSE {n3_inf := 0; n3_sup := N3-1};

FOR (i=n1_inf; i<=n1_sup; i++)
    FOR (j=n2_inf; j<=n2_sup; j++)
        FOR (z=n1_inf; z<=n1_sup; z++)
            CR[i+j*N1+z*N1*N2] := 1;
FIN;
```

Tema 6.3.4: Mantenimiento Hash

- El encadenamiento a cubo proporciona **extensiones automáticas** de espacio para cada dirección con poco coste.
- Por tanto, definir ELD para inserción no es (en general) una ventaja. Sí puede ser ventajoso tener una N más grande de lo necesario (para reducir la ocupación de los cubos, y demorar desbordamientos).
- Si el área de datos está demasiado saturada o demasiado vacía, o cuando la dispersión no es buena, o si las cadenas de cubos de cada dirección presentan demasiados huecos → es necesario **reorganizar**
- La reorganización automática no suele ofrecer buen rendimiento.

Lect. complementarias

- Una alternativa automática es truncar la dir. a los *x* últimos dígitos. Si tenemos que cambiar N, tomamos un dígito más o un dígito menos. → *Dispersión Extensible*
- Con un directorio se pueden evitar cubos vacíos → *Dir. Virtual*
- Si el directorio es arbóreo, la org. Es auto-extensible → *Dir. Dinámico*

Tema 6.3.5: COSTES Direccionamiento

Actualización:

- inserción: **2** accesos
- borrado (de *k* registros): ***selección*** + ***k*** accesos
- modificación (de *k* registros):
  - no se modifica la CD: ***selección*** + ***k*** accesos
  - se actualiza la CD: borrado y reinserción (***selección*** + **3·*k*** accesos)

Recuperación: el coste de la selección (***selección*** accesos)

Coste de Seleccionar Registros (por CD)**:**

- selección identificativa: $\mathbf{1 + P_{desb} \cdot (N_{desb}+1)/2}$ **accesos,,** N_{desb} { **- N área desb.** / **- N en cadena** / ***(long. cadena)*** }
- consulta selectiva no identificativa: $\mathbf{1 + N_{desb}}$ **accesos**
- consulta selectiva multiclave: $\mathbf{2^q + N_{desb}}$ **accesos,,** $q = \sum \log_2(N_i)$,, $\forall$ CD_i ausente
- consulta por clave alternativa: *full scan* = $\mathbf{N + N_{desb}}$ (cubos en área desb.)

Tema 6.4: Procesos de Recuperación

- Lectura y Filtrado: *full scan* vs. acceso aleatorio (cl. privilegiada)
- Ordenación: tablas pequeñas (quicksort) vs grandes (mergesort)
- Agrupación: dispersión de la tabla por el criterio de agrupación
- Combinación de dos orígenes de datos (*join*)
  - recorridos anidados: se leen filas en dos bucles anidados;
    - para combinación: mantiene la concatenación de filas que cumplen la condición de combinación (si existe)
    - para filtrado: mantiene las filas de la maestra (conductora) que cumplen una condición sobre la segunda (conducida)
    - condiciones de parada: las condiciones 'exists' / 'not exists' así como las condiciones 'in' / 'not in' pueden detener el recorrido de la segunda tabla al hallar el primer resultado

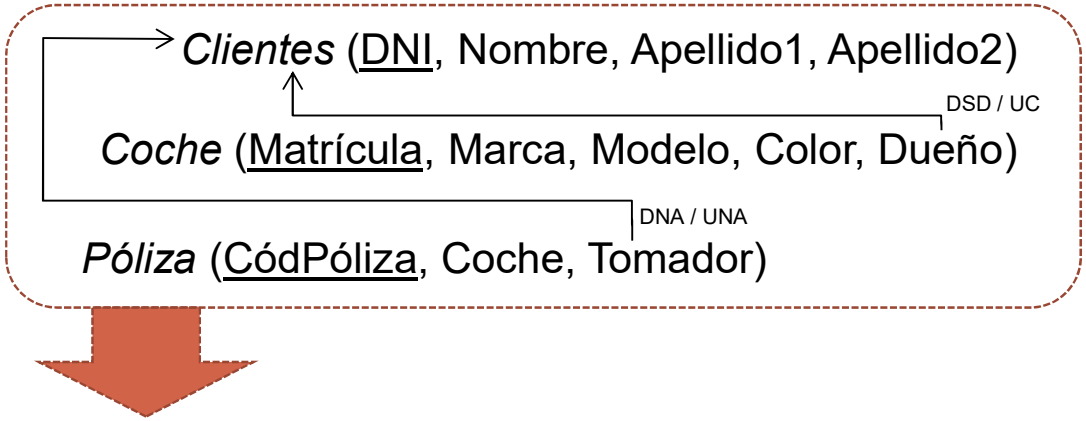
Tema 6.4: Procesos de Recuperación

- Combinación de dos orígenes de datos (*join*)
  - composición dispersa: si una de las tablas no es muy grande, se puede realizar una dispersión en memoria de una de las tablas (la menor) y hacer después un recorrido serial de la otra tabla
  - orígenes ordenados: es la opción más eficiente si ambas tablas son grandes. En primer lugar, se requiere ordenar ambas tablas por clave de combinación; después, iterativamente, se leerán las cabeceras de ambas secuencias y se combinan si son iguales (si no lo son, se descarta la menor).
    - Admite variantes con varias secuencias para uno de los orígenes o los dos (mezcla y combinación simultáneas)
  - Incondicional (cartesiana): intentará mantener una de las tablas (o un fragmento) en el buffer; habitualmente costoso (salvo excepciones: una tabla muy pequeña, poca proyección,…)

Tema 6.4: CLUSTER de datos (grupo)

- Consiste en almacenar físicamente juntos (en la misma celda o cubo) todos los registros que tengan el mismo valor para una clave privilegiada (clave de agr. físico o *clusterización*).
- Un archivo o varios en el mismo cluster.

Ejemplo:




```
CLUSTER identidad
( DNI C(9),
  cliente (nombre C(25), apellido1 C(15), apellido2 C(15)),
  coche (matrícula C(7), marca C(20), modelo C(20),color C(10) )*,
  poliza (cod C(30), coche C(7) )*
);
```

Tema 6.4: CLUSTER de datos (grupo)

- Si en el *cluster* se integran varias tablas, cada dirección contendrá registros de distinto tipo con un atributo común
  → *se define un nuevo registro (global)*
- La agrupación física *cluster* es una organización no consecutiva, y podría seguir cualquiera de las organizaciones de archivo:
- **Cluster Simple (serial):**
  - favorece la combinación de registros (… *clientes join coches* …)
  - puede mejorar la agrupación lógica (… group by DNI …)
  - … pero localizar es un proceso pesado (recorrido serial)
  - … y todo requiere localización (hasta insertar)
  - … y leer un archivo (select * from clientes) implica leer varios

Tema 6.4: CLUSTER de datos (grupo)

- Se persigue mejorar la localización (para inserción y selección).
- La clave privilegiada será la clave de clusterización

- **Cluster ordenado:** (CO $\subseteq$ CA)
  - puede mejorar algunos procesos selectivos y pr. ordenados
- **Cluster disperso:** (CD $\subseteq$ CA)
  - mejora la inserción y los procesos selectivos (por CD)
  - conserva mejoras en combinación y agrupación
  - …pero arrastra los puntos débiles de la *organización direccionada*
    - baja **densidad**: perjudica otros procesos (no privilegiados)
    - **desbordamientos**: políticas seriales vs. encadenamientos
    - no da buena respuesta a procesos ordenados
    - poca eficiencia en selecciones sobre **rangos**.

Tema 6.4: CLUSTER de datos (grupo)

- mejorando el **Cluster disperso:** (CD $\subseteq$ CA)
  - baja **densidad**: equilibrio entre espacio asignado a cada dirección y espacio de direccionamiento; puede interesar definir *celdas*
    - *celda:* espacio de cubo más pequeño que el bloque.
  - **desbordamientos**: políticas híbridas → extensiones encadenadas (varios cubos seriales consecutivos asignados a la vez).
  - los registros de cada dirección (cubo) pueden mantenerse ordenados (CO $\subseteq$ CA) para acelerar procesos de ordenación por mezcla natural.
- **Cluster indizado:** (CI $\subseteq$ CA)
  - requiere almacenamiento auxiliar y accesos extra
  - mantiene gran parte de las ventajas, y ofrece mejor respuesta en selecciones sobre **rangos**

Tema 6.4: CLUSTER en Oracle®

- El cluster garantiza que **toda la fila** combinada (el resultado del join de todas las tablas implicadas para un valor del cluster) se almacena físicamente **en el mismo cubo**
- Permite definir el tamaño de celda (por defecto, un cubo), incluso menor que el tamaño de bloque; tiene que ser potencia de 2.
- En Oracle®, un *cluster* puede ser **indizado** o **disperso**
  (también admite cluster **disperso ordenado**, agilizando mezcla natural)

Ejemplo:

```
CREATE  CLUSTER identidad (DNI VARCHAR2(9));
CREATE  TABLE cliente(…) CLUSTER identidad (DNI);
CREATE  TABLE coche(…) CLUSTER identidad (dueño);
CREATE  TABLE poliza(…) CLUSTER identidad (tomador);
CREATE INDEX ind_dni ON CLUSTER identidad;
```