Tema 7: Organización de Ficheros: Organizaciones Auxiliares (I)

- **Introducción**
  - **Concepto de Índice y Apuntamiento**
  - **Diseño básico de índices**
  - **Operaciones y Coste de Procesos sobre Org. Indizadas**
  - **Ventajas e inconvenientes de su aplicación**
- **Taxonomías de Índices: simples e Índices Multinivel**
- **Indización por Árboles B**

Tema 7.1: Concepto de Índice

Las organizaciones base suelen establecerse entorno a un proceso privilegiado (o unos pocos procesos).

Serial → privilegia inserciones
Secuencial → privilegia algún acceso ordenado (y alguna localización)
Direccionada → privilegia localizaciones a través de una clave

- El resto de procesos selectivos (clave alternativa) son pesados (*full scan*)
- Si una cl. alternativa es muy frecuente, se puede almacenar en un archivo aparte la ubicación física de cada valor de esa clave.

  Ejemplo: en un libro, la clave *título_capítulo* se almacena asociada a la ubicación del registro (núm. página) en un archivo aparte (**índice**).

Tipo de archivo auxiliar: *índice* (directorio)

Clave privilegiada: *clave de indización*

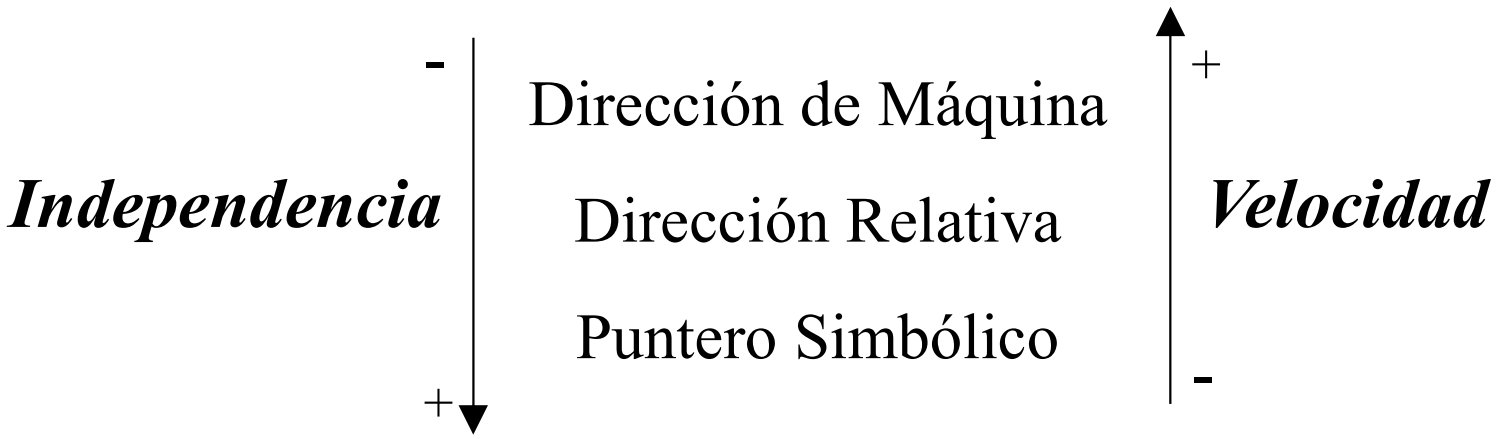
Por ser auxiliar, se pueden establecer cuantos índices se estime oportuno

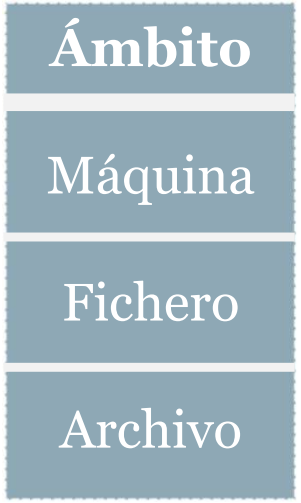
Tema 7.1: Conceptos previos

Tipos de Puntero (según su dominio):

- *Dirección de Máquina*: la dirección física del registro
- *Dirección Relativa*: del registro en el espacio de dircmto. del fichero
- *Puntero Simbólico* (identificador): identificación lógica del registro

(*Puntero Simbólico* no identificador: caracterización lógica de un cjto. de registros)





Tema 7.1: Concepto de Índice

- *Entrada*: registro formado por punteros
- *Directorio*: archivo formado por entradas (y por ende, por punteros)

  Algunos usos:- virtualizar direccionamientos (ver lect. comp. 6)
  - asociar registros (p.e., personas con sus coches)
  - traducir punteros

***ÍNDICE*: directorio cuya entrada se refiere a un solo registro**

- Es como un listado para traducir punteros (lógicos a relativos)

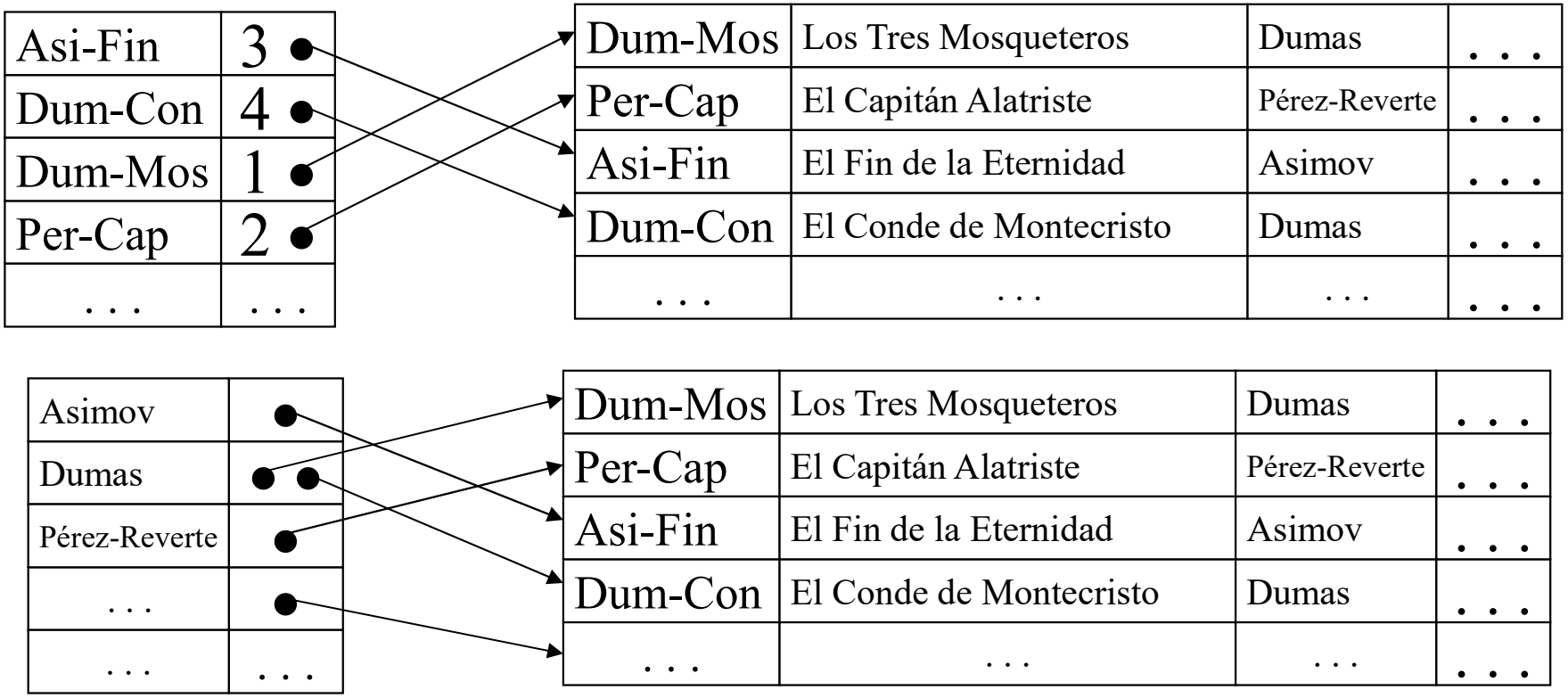
- *"Almacenamiento auxiliar utilizado para localizar los registros"*
  - Los índices se almacenan en un fichero (o varios) de índices.
  - Los registros, en el fichero de datos manteniendo su *organización base*

- ***La clave siempre es un puntero lógico*** *(no necesariamente unívoco)*
- ***El otro puntero suele ser relativo (dir. del cubo / posición en el cubo)***
  - Si el uso del índice es el filtrado (del cjto. dir. relevantes) → sólo parte alta ptro.
  - Si la org. base es virtual, se utilizará la dirección virtual

Tema 7.1.2: Tipos básicos de Índice

- **PRIMARIO**: la clave de indización es **identificativa**

  1 registro ↔ 1 valor de clave ↔ 1 entrada (1 puntero)



- **SECUNDARIO**: la clave de indización es **no identificativa**

  n registros ↔ 1 valor de clave ↔ 1 entrada (n punteros)

Tema 7.1.2: Diseño de la Entrada

- **Diseño Físico-Lógico de la entrada de índice Primario**:

  ENTRADA ≡ clave · puntero_externo

- **Diseño Físico-Lógico de la entrada de índice Secundario**:

  - Habrá varios registros con el mismo valor de clave...
  - Se almacena **sólo una vez el valor** de cada clave, con todos sus punteros.

  ENTRADA ≡ clave · long_lista · (puntero_externo)$^{long\_lista}$

  - Corolarios:
    - El número de entradas es igual a la card. del dominio: $e = \#valores(CI)$
    - Al buscar, sólo hay que recorrer el índice hasta encontrar una entrada
    - Para insertar en listas de punteros hay que buscar la entrada correspondiente y en caso de que no exista (tras recorrer todo el fichero) insertar al final.
      → Con listas, conviene tener el **índice ordenado,** y siempre **no consecutivo**
    - La longitud media de la lista es la coincidencia de clave $k = r / \#valores(CI)$

Tema 7.1.2: Operaciones sobre Ficheros Indizados

- Operaciones de administración / mantenimiento:
  - ***Creación***: hay que crear el índice (al crear el fichero o posteriormente)
  - ***Borrado***: si se borra el fichero de datos, hay que borrar el índice
    Además, se puede destruir el índice sin borrar el fichero de datos.
  - *Reorganización*: si degenera, deberá reorganizarse periódicamente.
- Operaciones selectivas (localización de registros):
  - ***Existencia***: acceso al fichero de índices
  - ***Localización***: acceso al fichero de índices + acceso al fichero de datos
  - ***Consulta a la totalidad***: generalmente, no se utiliza el índice
    (excepción: proceso ordenado / índice ordenado / fichero no ordenado)
- Operaciones de Actualización:
  - **(1/2) Selección** (por CI): se puede hacer a través del índice
  - **(2/2) Actualización**: escritura fichero datos + escritura fichero índice

Tema 7.1.2: Coste de Procesos Ficheros Indizados

- **Localización a través del Índice:** acceso al índice (según su naturaleza)
- **Localización por varios índices:** suma del acceso a cada índice
- **Recuperación:**

$$C(O_i,P_j) = \text{acceso\_índice} + \text{acceso\_datos}$$

$$\text{acceso\_datos} = \#\text{cubos} \cdot E_c$$

- **Actualización:**
  - Inserciones: suelen requerir inserción de entradas — $\Delta\text{coste} = \text{acc\_índice} + 1$
  - Borrados: pueden localizarse con el índice — $\Delta\text{coste} = \text{acc\_índice} + 1$
    - índice primario: suelen requerir borrado de entradas
    - índice secundario: pueden requerir modificación de entradas
  - Modificaciones: pueden localizarse con el índice
    - CI: suele implicar borrado + reinserción de entrada — $\Delta\text{coste} = 2 \cdot \text{acc\_índ} + 2$
    - CD/CO: cambia ubicación reg. → cambia puntero — $\Delta\text{coste} = \text{acc\_índice} + 1$

Tema 7.1.3: Efectos de Indizar

Ventajas

1. *Acceso por **Claves Alternativas***
   Se gana eficiencia en la localización por claves (hasta ahora) no privilegiadas

2. *Aumento de la **Tasa de Acierto** (en M_{int})*
   El índice tiene menos cubos y de acceso más recurrente. Con buena gestión de M_{int}, el acierto es muy elevado en los accesos al índice (que son la mayor parte del acceso indizado) haciendo que se dispare la tasa de acierto global.

3. *Reorganización Menos Costosa*
   Ya que los índices tienen menos bloques que el f. de datos, este coste es menor

Inconvenientes

1. ***Procesos de Actualización Más Costosos***
2. *Necesidad de Almacenamiento Auxiliar*
3. *Necesidad de Operaciones de Mantenimiento*

Tema 7.2: Ficheros Indizados.

Taxonomías de Índices

- Según el carácter (identificativo) de la clave de indización:
  - ***índices primarios vs. índices secundarios***

- Según la correspondencia (biyectiva o no) entre entradas y registros:
  - ***Denso*** (1:1): existe una entrada del índice para cada registro
  - ***No Denso*** (1:n): una entrada para cada cubo de datos

- Según el recubrimiento del índice:
  - ***Exhaustivo***: todos los registros que deben tener entrada la tienen
  - ***Parcial***: no se indizan todos los registros

  (se dejan aparte los que se acceden rara vez, los últimos en ser introducidos, etc.)

  → Si el índice parcial falla (∅), no aporta ningún valor informativo (no filtra).

- *Según la estructura: índices simples vs. índices multinivel (arbóreos)*

Tema 7.2: Índice Simple Denso

- **Naturaleza**: serial, secuencial, o direccionado

- **Coste**: dependiendo de su naturaleza (igual que un fichero de datos análogo)

- **Restricciones**: se debe **aplicar sobre claves no privilegiadas**

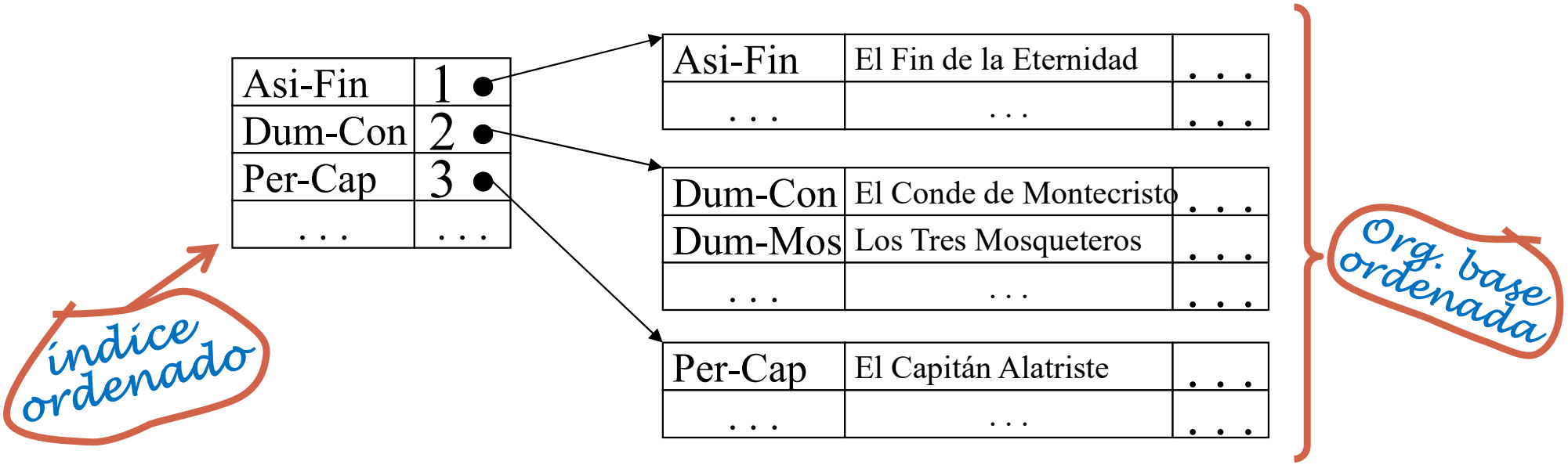
Ejemplos:

- si el índice se usa para un proceso ordenado por CI (y además CI≠CO)
- si la CO se usa para un proceso ordenado y la CI para procesos selectivos
- si el índice se usa para un proceso especial (ver acceso invertido, tema 7.5)

Mantenimiento:

- si es ordenado o disperso, puede desbordar → requiere reorganización
- debe evitarse la degeneración de la estructura
  - → índice ordenado: preferible inserción ordenada + reorganización local
  - → índice disperso: pierde eficiencia si cambia (si es volátil)
    es más útil como índice temporal (procesos puntuales)

Tema 7.2: Índice Simple No Denso

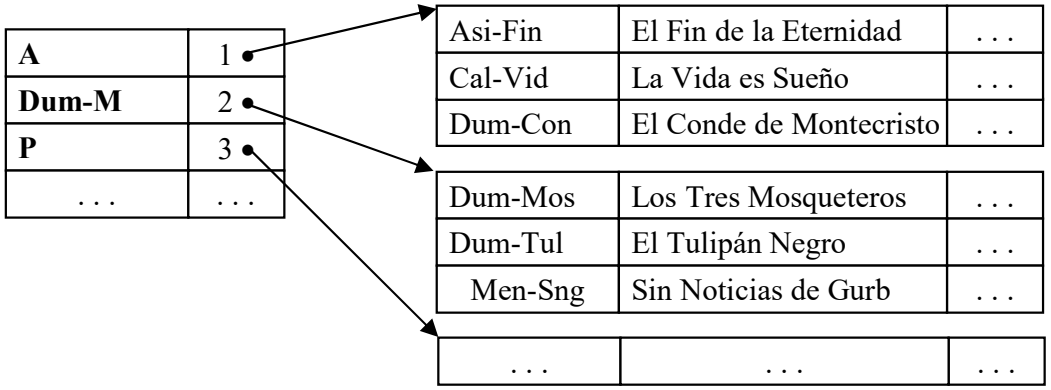


- **Concepto**: una entrada por cubo de datos (en lugar de una entrada por registro)
- **Restricción**: índice y organización base deben ser necesariamente secuenciales y con ***clave_indización = clave_ordenación (CO=CI)***
- **Usos**: aporta varias posibilidades de acceso:
  - procesos ordenados (a la totalidad): acceso serial de la org. base (ordenada)
  - procesos selectivos (solución única): a través del índice
  - mixtos (selección de un rango): acc. indizado (1er elemento) + serial

Tema 7.2: Índice Simple No Denso

Ventajas:

- al ser de tamaño muy reducido, tiene menor coste (y mayor tasa de acierto)
- se ahorran muchas actualizaciones de índice (insertar/borrar/modificar registros a menudo no afectan al índice, salvo que sean el primer registro del cubo).
- en lugar de utilizar toda la *clave* en la entrada, se pueden usar prefijos (%tamaño)

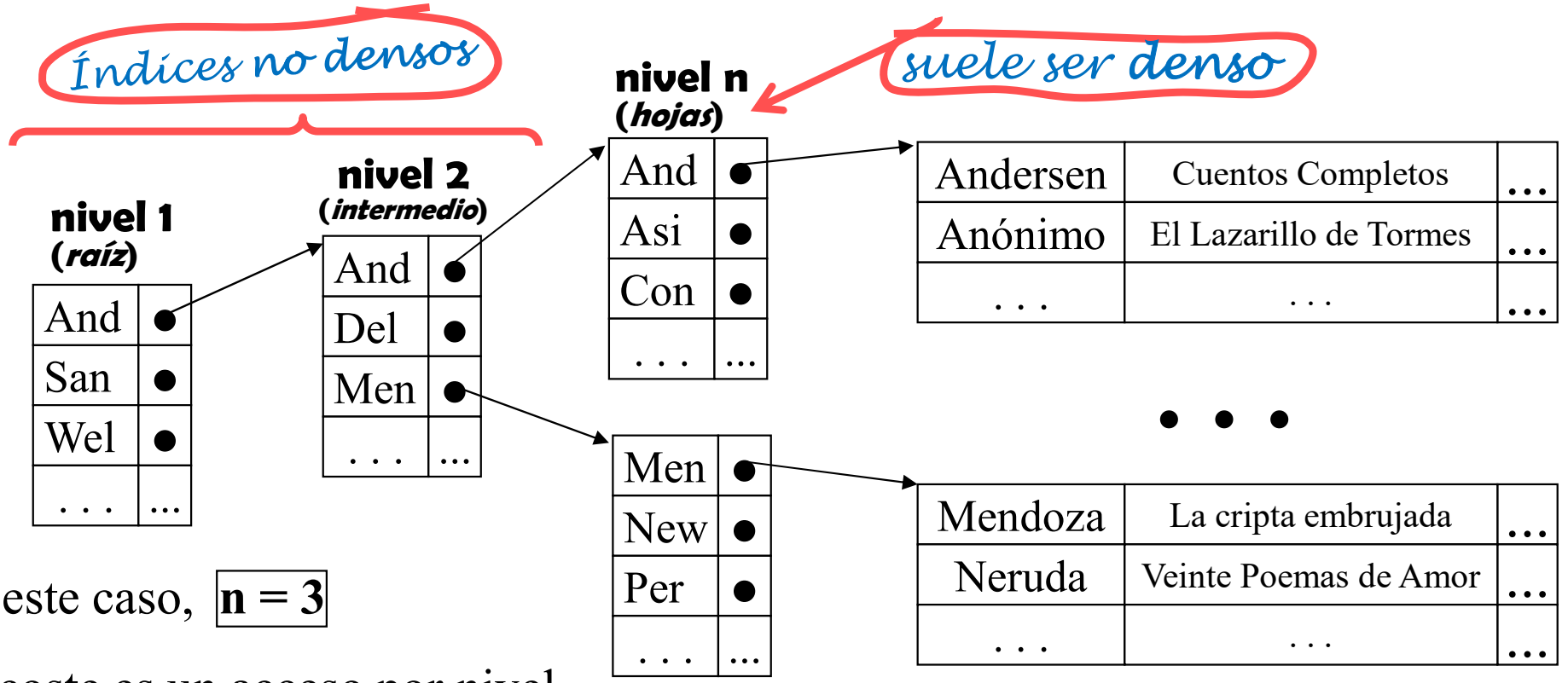


Inconvenientes:

- sólo puede existir un índice no denso para cada archivo
- la inserción del registro debe ser ordenada, pero se localiza con el índice.
- La inserción de la entrada es ordenada, y conlleva pesadas reorganizaciones
  - deben aplicarse mecanismos de ELD, rotación, partición celular, etc.

Tema 7.2: Índice Multinivel

Concepto: es un índice con *n* niveles (el nivel n es índice del nivel n+1)



- En este caso, **n = 3**
- El coste es un acceso por nivel
  Interesa definir **nodos pequeños** (1 bloque) incluso a costa de tener más niveles
- Es ventajoso **bloquear la raíz** (nivel 1) en memoria intermedia (ahorra un acceso)
- El nivel n suele ser denso, pero como es secuencial puede ser un índice no denso

Tema 7.2: Índice Multinivel

- El índice multinivel perfectamente construido es eficiente, pero degenera
- En ficheros constantes es buena solución
- En ff. volátiles se requiere reorganización local → evolución a otras estructuras

Árboles Binarios: cada nodo es una entrada del índice con dos punteros internos
Solución sencilla, pero presenta problemas de vecindad y desequilibrio.

- **Árbol AVL**: resuelve desequilibrio mediante procesos de reorganización local
- **Árboles Binarios Paginados**: resuelven el problema de vecindad
- **Árboles AVL-Paginados**: buen rendimiento, pero necesitan muchos punteros internos, presentan bajísima densidad, y reorganizaciones frecuentes.

Solución:

- *Incluir varias entradas por nodo (y, por tanto, varios descendientes)*
- *Construir el árbol en orden ascendente (el separador pertenece al nodo que desborda)*

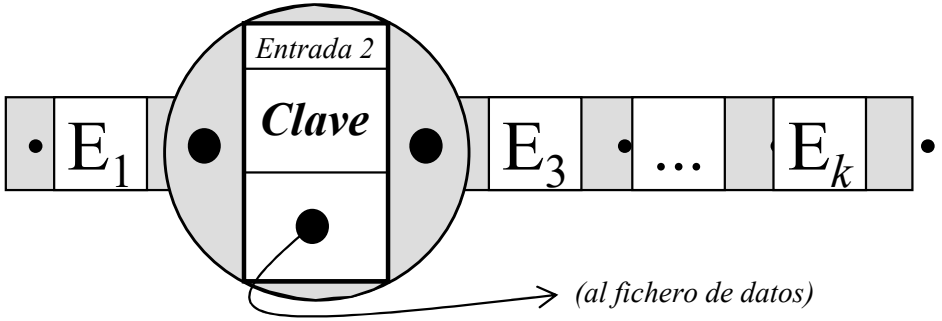
} **árboles B**

Tema 7.3.1: Indización en Árbol B

- *Propuestos por Bayer y McCreight*

Idea: ya que inicialmente no se conoce el elemento que es mejor separador, se comienza por las hojas. A medida que crezca, se construye hacia arriba.

- **Nodo**: cada nodo va a llenar la página; contiene entradas de índice (pares *clave indización-puntero a los datos*) y punteros (para apuntar nodos hijo)



- **Orden del árbol**: indica la capacidad de los nodos (y por ende, del árbol)
  - según las entradas: el nº mínimo de claves de un nodo (*Bayer*)
  - según los punteros (hijos): nº máximo de hijos de un nodo (*Knuth*)

Tema 7.3.1: Indización en Árbol B

Árboles B : Observaciones

- Si el árbol es de orden ***m***, cualquier nodo tendrá a lo sumo *m* descendientes
- Si un nodo tiene *m* descendientes (no hoja), tendrá m-1 entradas
- Corolario: un nodo de un árbol de orden *m* tiene a lo sumo ***k*=*m*-1** entradas
- En un nodo (T_{nodo} bytes) caben ***m*** punteros internos y ***k*** entradas, luego:

$$m \cdot T_{ptro\_interno} + k \cdot T_{entrada} \leq T_{nodo}$$

(con esta fórmula y $k=m-1$ se podrá hallar el orden del árbol)

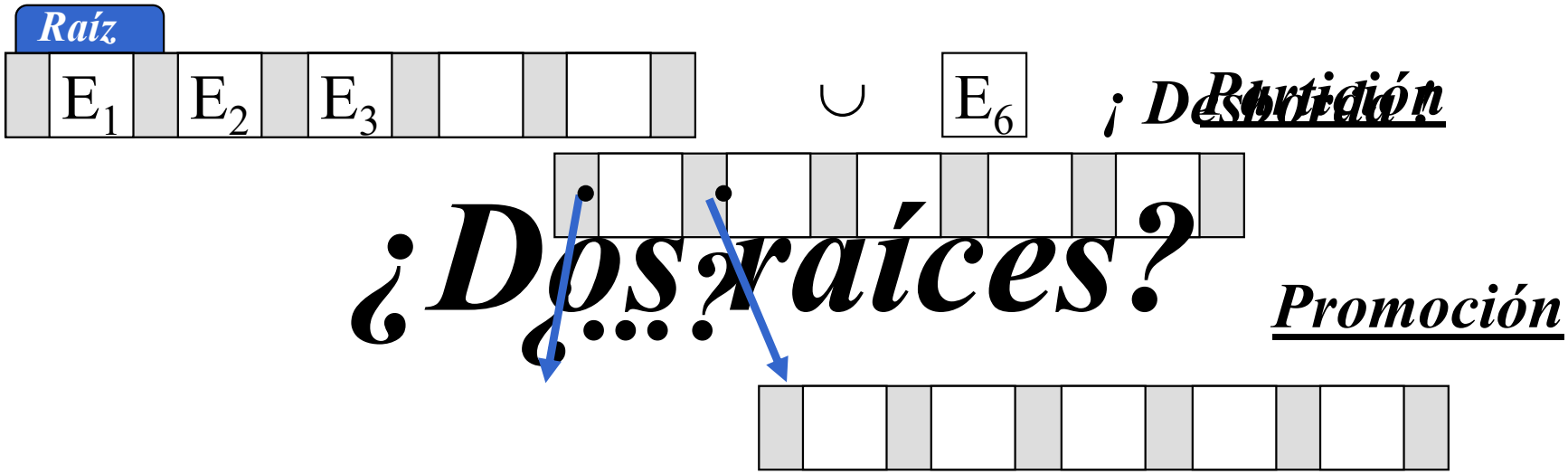
- El nodo raíz tiene al menos un elemento y, por tanto, al menos 2 hijos.

- $\mathbf{T_{nodo}}$ es múltiplo de T_{bloque} y suele ser lo menor posible (**típicamente 1 bq**)
- $\mathbf{T_{entrada}}$ es la suma del tamaño real de la clave (fija/marcada/codificada…) más el/los puntero/s interno/s (pueden ser muchos punteros, si es secundario)

Tema 7.3.1: Indización en Árbol B

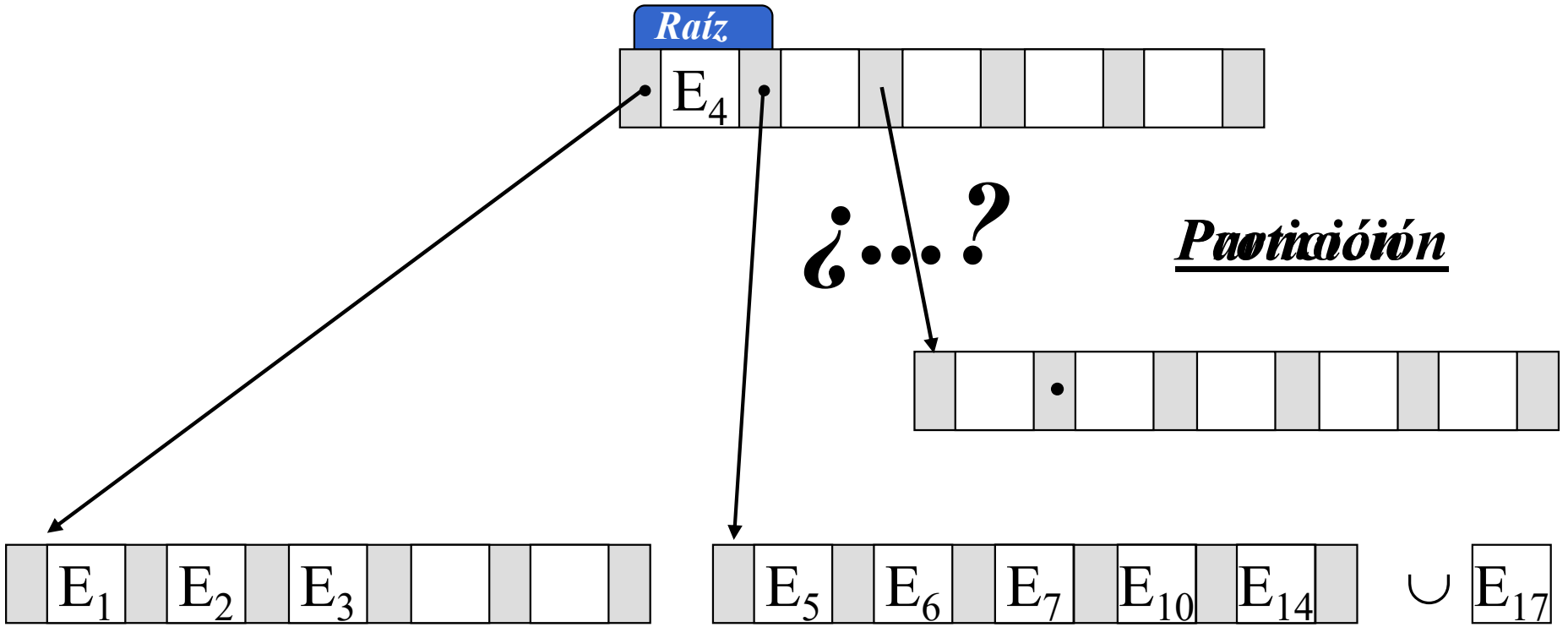
Árboles B : Partición y Promoción

- Las entradas dentro de un nodo van ordenadas
- Cuando un nodo *desborda*, se divide en dos y se promociona el elemento intermedio hacia el nivel superior (ese elemento se lleva dos punteros: uno hacia cada hijo, es decir, hacia cada uno de esos dos nuevos nodos)



Tema 7.3.1: Indización en Árbol B

Árboles B : Más Partición y Promoción (otro ejemplo)



Tema 7.3.1: Indización en Árbol B

Árboles B : Propiedades

- Todos los nodos menos el raíz garantizan una **ocupación mínima**: $k_{mín} = \lfloor \frac{k}{2} \rfloor$

Corolario

- ¿Cuántos descendientes como mínimo tienen los nodos intermedios?

  (suponiendo política de *'dividir cuando desborda'*)

  $m_{mín} = k_{mín} + 1$ → $m_{mín} = \lfloor \frac{m+1}{2} \rfloor$

- Tamaño del fichero de índices

Se puede obtener una cota superior del fichero de índices

$N^o_{máx}$ ***nodos fichero*** = nº entradas fichero / $k_{mín}$

$T_{máx}$ ***fichero*** = $n^o_{máx}$ nodos fichero · T_{nodo}

Tema 7.3.1: Indización en Árbol B

Árboles B : Propiedades (II)

El nº de niveles (n) para un árbol de orden m y e entradas tiene **cota superior**

nivel	nodos	entradas	acumulado

Cota Superior:

$$e \leq 1 + log_{\lfloor \frac{m+1}{2} \rfloor} (\frac{e+1}{2}) \geq n$$

$> e$

(este nivel es imposible)

Tema 7.3.1: Indización en Árbol B

- Para **recuperar una entrada**: #accesos = #niveles
  - Dado que la raíz estará siempre en memoria, contamos un acceso menos
- Para **recuperar un registro aleatorio** , se recuperan la entrada y tantos cubos de datos como punteros tenga la entrada (es decir, k cubos)

$$C(O_i,P_j) = (n\text{-}1)\cdot T_{nodo} + c \cdot E_c$$

- El coste de cualquier actualización sobre el índice en árbol B es el coste de **localización más un acceso** de escritura: $\Delta C(\text{actualización}) = (n\text{-}1) + 1 = \boldsymbol{n}$
- El coste extra de una partición es de **dos accesos** de escritura (actualizar el nodo antecesor y escribir el nodo nuevo).
- El tiempo de acceso así calculado es una *cota superior al tiempo de acceso.*
- Puede calcularse la cota inferior (en base al número de niveles del árbol perfectamente construido), para conocer su coste óptimo y valorar el beneficio de ejecutar la reorganización del índice.

Tema 7.3.1: Indización en Árbol B

Árboles B : Valoración

Aspectos Positivos:

- Existe una cota superior razonable del número de accesos a soporte
- Generalmente, las operaciones (de inserción o borrado) requieren reestructurar una página. Y si son más, suelen ser pocas páginas.
- En el peor caso, las páginas están ocupadas a la mitad (aproximadamente)

Aspectos a Mejorar

- Si las entradas son grandes, el orden puede ser demasiado pequeño
- La densidad (mínima) de los nodos es muy mejorable
- En las hojas se desperdicia mucho espacio (no necesitan punteros)

Tema 7.3.2: Indización en Árbol B*

Idea: se pretende **aumentar la densidad** de los nodos

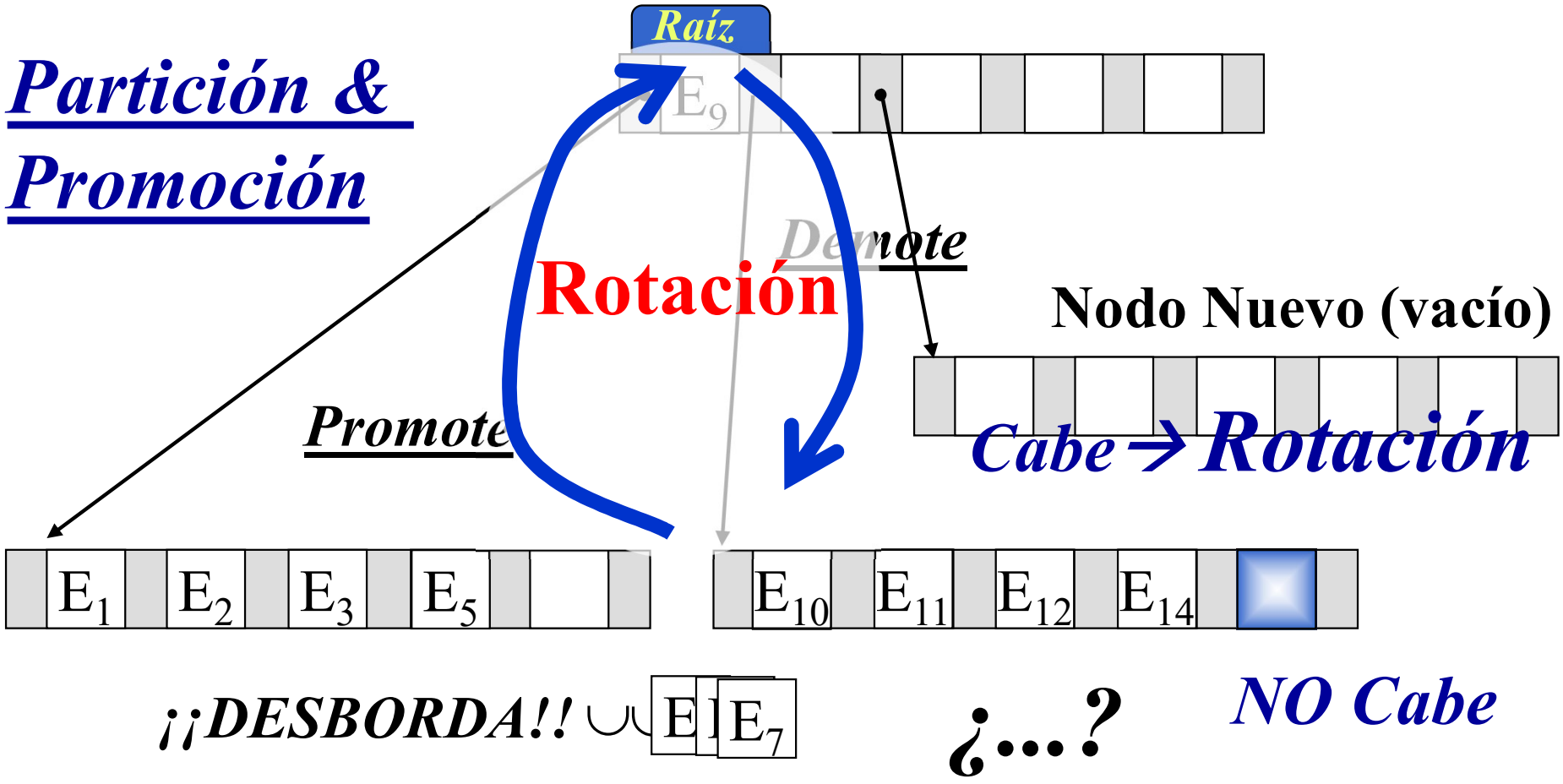
Para ello, en lugar de dividir un nodo en dos, se dividirán dos nodos en tres.
Así, en lugar de conseguir una ocupación mínima del 50% se obtendrá el 66%

- Cuando un nodo desborda, en lugar de dividir, se procurará ceder uno de sus elementos a su vecino (**rotación**).
- Si el nodo vecino también está lleno, se parte (dos nodos llenos en tres nodos)
- Por lo demás, el resto del funcionamiento es como el de los árboles B.
- **Ventajas**:
  - *Aumento de la densidad (al 66%)*
  - *Un desbordamiento no siempre supone partición/promoción*
- **Desventajas**:
  - *Aumenta la probabilidad de desbordamiento (nodos más llenos)*

Tema 7.3.2: Indización en Árbol B*

Rotación, Partición y Promoción (ejemplo)

Partición & Promoción



Tema 7.3.2: Indización en Árbol B*

Propiedades

- Todos los nodos menos el raíz garantizan una ocupación mínima: $k_{mín} = \lfloor \frac{2k}{3} \rfloor$
- Los nodos intermedios cuentan con $\frac{2k}{3}+1$ descendientes $\rightarrow$ $m_{mín} = \lfloor \frac{2m+1}{3} \rfloor$

  (suponiendo política de *'dividir cuando desborda'*)

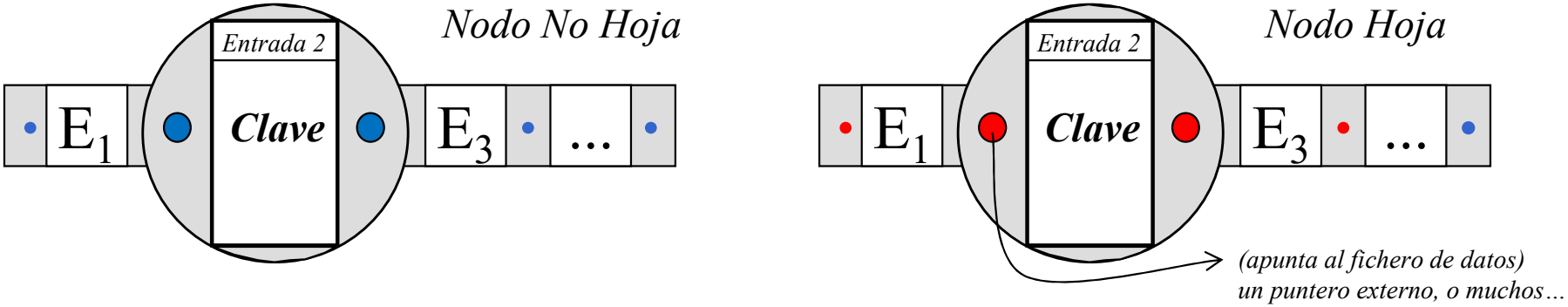
Cálculo de Costes

- Como la localización es idéntica al árbol B, también es igual el cálculo de costes
- El coste extra de una rotación es de tres accesos (lectura del nodo contiguo, más la escritura de ese nodo y del nodo antecesor). ΔC(rotación) = 3 acc
- La partición implica cuatro accesos extra (la lectura del nodo contiguo, más la escritura de los nodos contiguo, nuevo, y el antecesor). ΔC(partición) = 4 acc
- También se puede contemplar la rotación bidireccional. En este caso la densidad es 75% ($k_{mín}=\lfloor 3k/4 \rfloor$) pero también el coste de inserción (rotación 4, y partición 5).

Tema 7.3.3: Indización en Árbol B+

Idea: coste proporcional a la profundidad → crecer en amplitud → aumentar el #hijos por nodo → **aumentar el *orden***

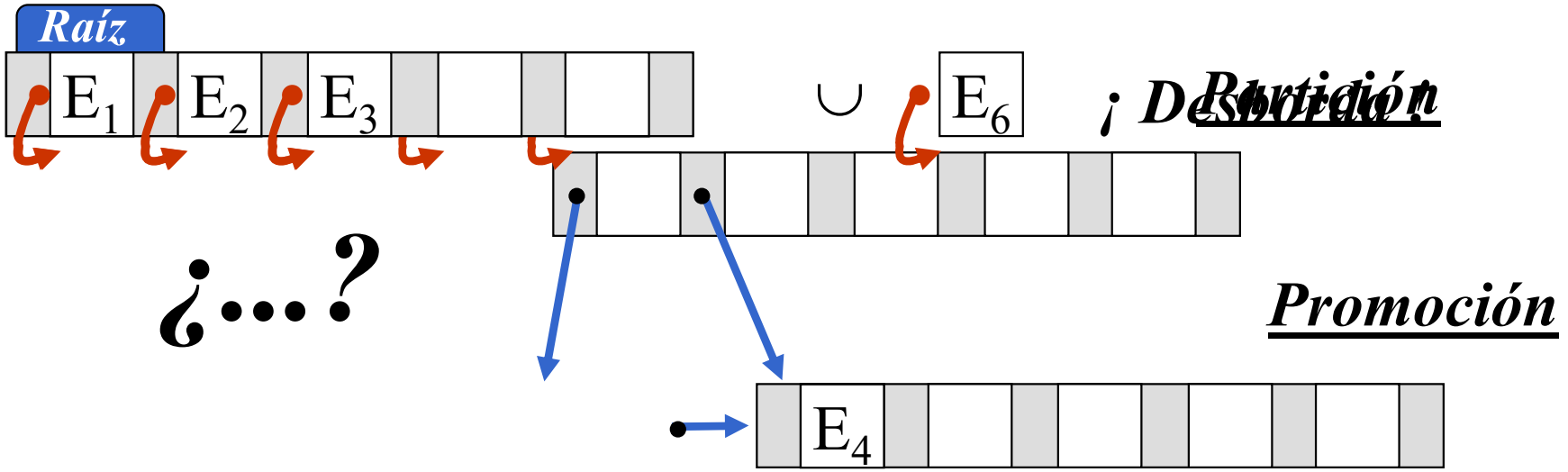
- En los nodos con hijos (nodos no hoja) **se suprimen los punteros externos** (así caben más discriminantes, y por ende se tienen más punteros internos).
- En los nodos hoja no hay punteros a nodo hijo, pero sí habrá punteros externos. Para apuntar a los datos, la entrada debe estar en una hoja → todas las entradas están en nodos hojas, y en los no hoja sólo hay copias discriminantes



- Especialmente eficiente con punteros externos grandes → **índice secundario**

Tema 7.3.3: Indización en Árbol B+
Partición y Promoción

Ejemplo de Partición y Promoción: nodo hoja/raíz *(primera partición)*

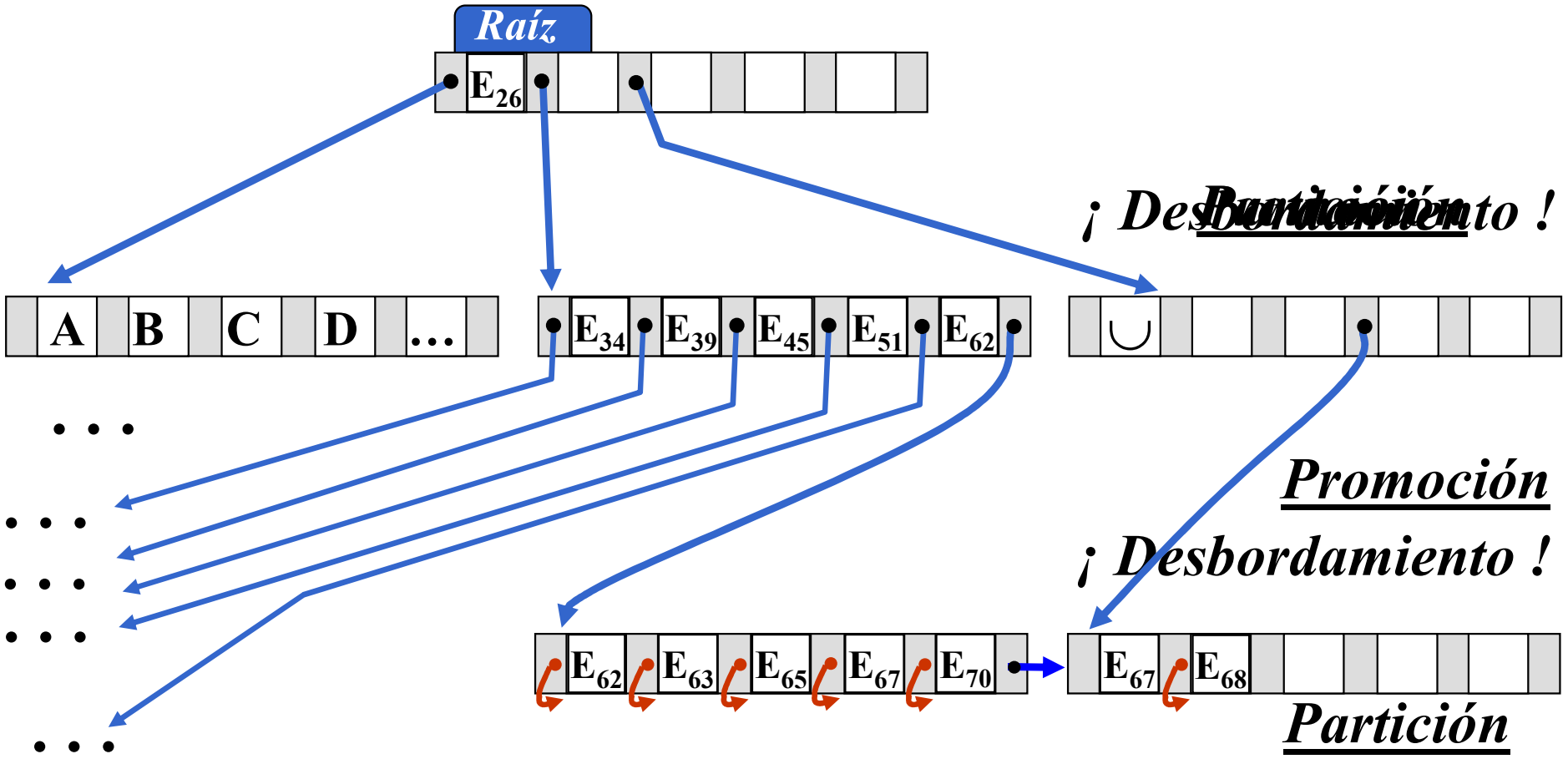


- Observar que en las hojas se usa un puntero interno adicional (puntero encadenamiento) para apuntar al siguiente nodo hoja (hermano). Esto se realiza durante la partición:
  - el ptro_encadenamiento(nodo_nuevo) := ptro_encadenamiento(nodo_viejo)
  - el ptro_encadenamiento(nodo_viejo) := dirección(nodo_nuevo)
- El ***encadenamiento de hojas*** proporciona un mecanismo de acceso alternativo.

Tema 7.3.3: Indización en Árbol B+
Partición y Promoción

- En nodos no hoja, la promoción es igual que en nodos de árbol B

Ejemplo:



Tema 7.3.3: Indización en Árbol B+

Propiedades (kmín y mmín)

- Orden del árbol (***m***): se calcula para nodos no hoja, como en árboles B; teniendo en cuenta que las entradas esos nodos carecen de puntero externo

$$m \cdot T_{puntero\_interno} + (m-1) \cdot T_{clave} \leqslant T_{nodo}$$

- Ocupación máxima (***k***) de los nodos hoja: si los tamaños de los punteros interno y externo son distintos, convendría calcularla por separado

$$k \cdot (T_{clave} + T_{puntero(s)\_externo}) + T_{puntero\_interno} \leqslant T_{nodo}$$

debe contener al menos una entrada comp.

encadmto. bidireccional req. dos punteros internos

- La ocupación mínima de las hojas será:
  (suponiendo política de *'dividir cuando desborda'*)

$$k_{mín} = \lfloor \frac{k+1}{2} \rfloor$$

- La ocupación mínima de los nodos intermedios será $\lfloor k/2 \rfloor$ → $m_{mín} = \lfloor \frac{m+1}{2} \rfloor$

  (la promoción en estos se opera como en los nodos de un árbol B)

Tema 7.3.3: Indización en Árbol B+
Profundidad y Tamaño

Cálculo del número de niveles:

- El nivel de las hojas es el **nivel *n***. ¿Cuántas hojas?

$$n^o\ hojas = \lfloor e\ /\ k_{mín} \rfloor$$

e = nº total de entradas

- El nº de nodos en el **nivel *n-1*** depende del número de nodos del nivel *n*

$$n^o\ nodos\ (n\text{-}1) = \lfloor n^o\ nodos\ (n)\ /\ m_{mín} \rfloor$$

- Cuando se llega a un nivel con un solo nodo (la raíz), este será el **nivel 1**.
  (se tiene que el nivel *n-x=1,* y se puede despejar *n = profundidad del árbol*)

- **Tamaño máximo** del fichero índice:
  se calcula como la suma de los nodos necesarios
  para cada nivel ($\sum_{i=1}^{n}$ nodos(i)) multiplicado por el tamaño de un nodo.

Tema 7.3.3: Indización en Árbol B+

Consideraciones finales

- El encadenamiento de hojas proporciona mecanismos de acceso alternativo.
- Ejemplos: (índice en árbol B^+ con clave indización '*fecha* (dd-MM-AAAA)')
  - procesos a la totalidad ordenados
    Ejemplo: sacar un listado de todos los registros ordenados cronológicamente
  - procesos selectivos con tasa de actividad elevada
    Ejemplo: recuperar todos los registros con fecha en mes de 'Mayo'
  - procesos ordenados con varios resultados (*rangos*) → acceso mixto
    Ejemplo: recuperar todos los registros entre el 01-05-2005 y el 30-06-2005
    - Los accesos mixtos consisten en recuperar la primera entrada (01-05-2005) a través del árbol, y el resto de entradas se recuperarán con el encadenamiento

- Se puede **organizar** un **fichero de datos en árbol**

  → es como tener un f. secuencial con part. celular y un índice B^+ no denso.

- Las mejoras logradas con árboles B^+ y B* son combinables