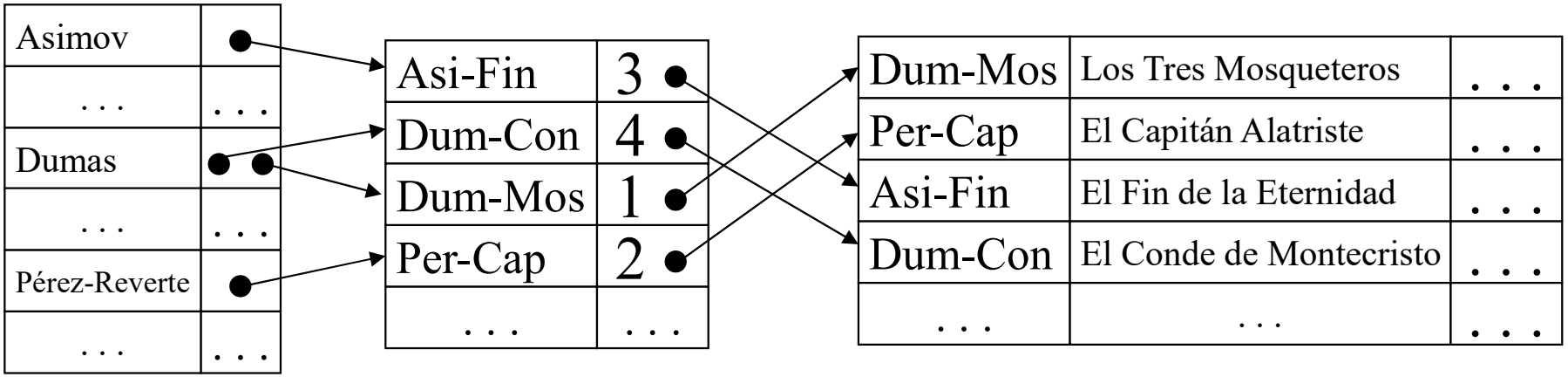
Tema 7': Organización de Ficheros: Organizaciones Auxiliares (II)

- **Estructuras Auxiliares Especiales**
  - **Índice intermedio**
  - **Índice agrupado**
  - **Índices Multiclave: árboles R**
  - **Índice BitMap**
- **Procesos sobre índices**
  - **Accesos al índice (simple/rango/alterno/total/rápido)**
  - **Conjunto de Direcciones Relevantes**
  - **Acceso Invertido**

Tema 7.4.1: Estructuras Especiales

Índice Intermedio

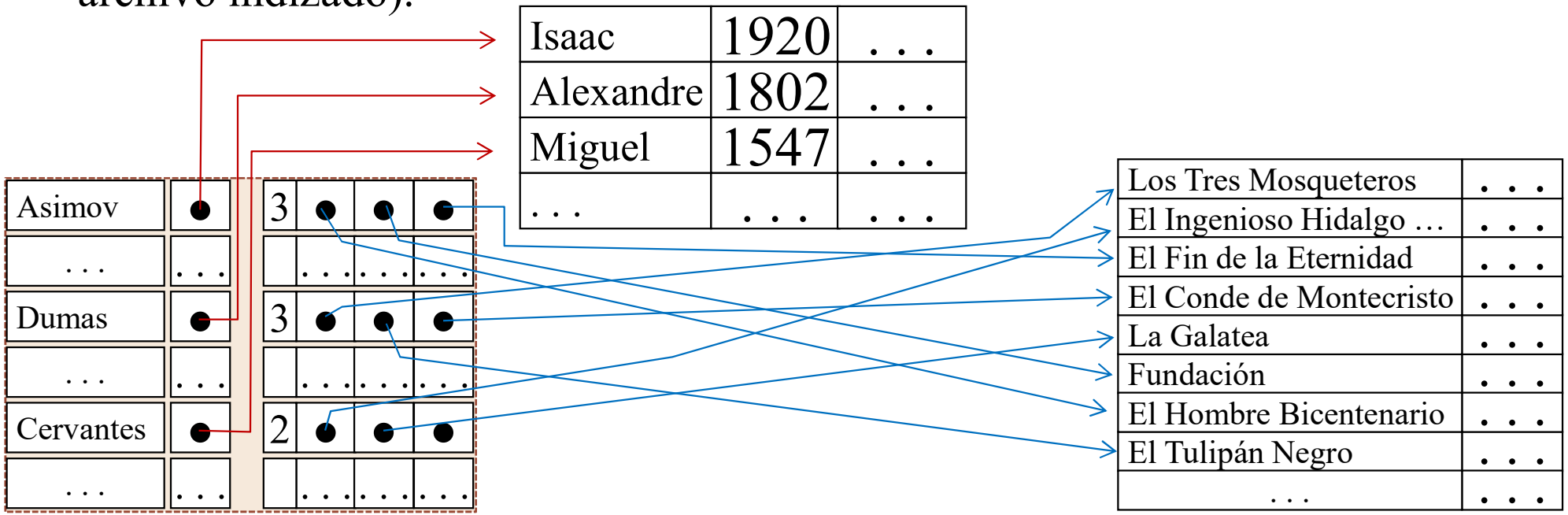
- Si los registros (en el f. de datos) cambian de ubicación, es necesario actualizar todos los índices de ese archivo.
- *Índice Intermedio*: índice primario cuyos punteros apuntan a los datos, y el resto de los índices apuntan a este. Al cambiar los registros de ubicación, sólo es necesario actualizar punteros en este índice.
- Este índice debe ser muy eficiente: bloqueado en memoria privilegiada de tamaño **reducido**, y (casi) **constante** (poco o nada volátil)



Tema 7.4.2: Estructuras Especiales

Índice Agrupado o Cluster

- Dos (o más) índices sobre distintos archivos con la misma CI y valores validados (por integridad referencial) pueden combinarse (***index join***).
- También puede crearse una estructura única de indexación (índ. *cluster*).
- La entrada tendrá una clave de indización y uno o más esquemas de punteros (un puntero o una lista de punteros, según sea la naturaleza de la clave en cada archivo indizado).

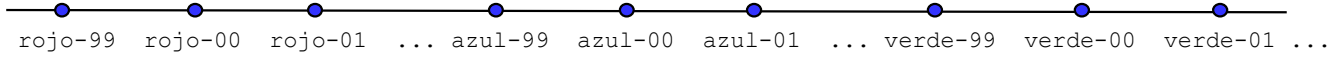


Tema 7.4.3: Estructuras Especiales

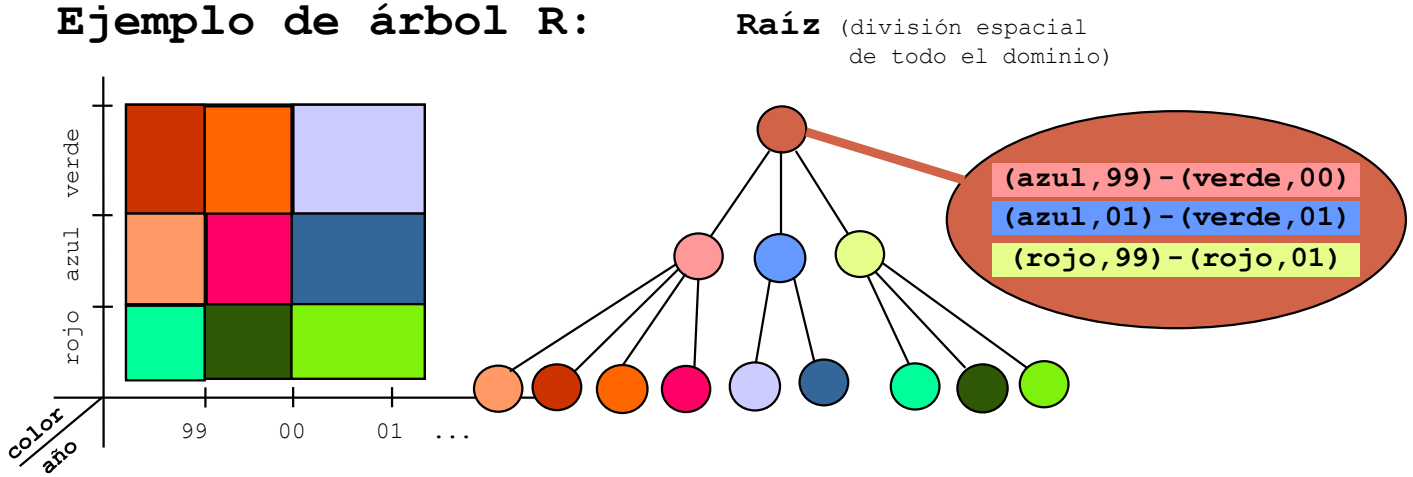
Índice Multiclave: el árbol R

- La indización multi-clave también admite la creación de índices especiales que no estén basados en una clave, sino en varias simultáneamente
- Es el caso del árbol R, una evolución del árbol B^+ para *d* dimensiones (claves). Propuesto por Güttman (1984)
- Cada entrada no es un punto en una línea, sino un intervalo d-dimensional:

Distribución lineal:



Ejemplo de árbol R:



- raíz con 2 e. o más (salvo si es hoja)
- altura balanceada
- los intervalos pueden solaparse (al buscar un intervalo hay que recorrer todos los descendientes que intersecan con el intervalo en cuestión)

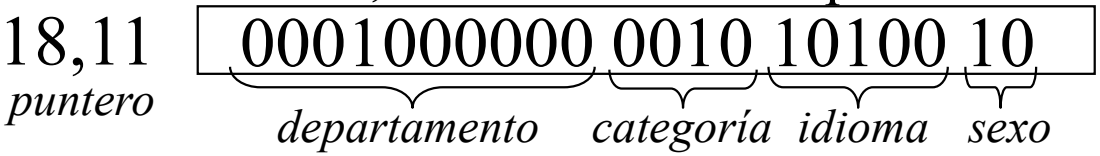
Tema 7.4.4: Estructuras Especiales

Esquemas de bits (BITMAP)

Un *esquema de bits* para un campo es un vector de valores booleanos.
A cada **valor** del dominio se le hace corresponder una **posición**.

Ejemplo: *idioma* (castellano, inglés, francés, alemán, italiano)

- Si la cardinalidad (#valores) es alta, puede indizarse un subconjunto (índ. parcial).
- Admite la multivaluación (varios valores para la misma entrada)
- Puede ser simple o multiclave, concatenando esquemas de varios campos



- Puede utilizarse como *directorio de ocurrencia*
- ***Proceso serial*** (excepción: acceso invertido sobre bitmap)

Tema 7.4.4: Estructuras Especiales

Máscaras sobre BITMAP

- **Máscaras para condiciones de igualdad:**
  - Se realizan con un bit para cada posible valor, en el conjunto { 0 , 1 }
  - La selección comprueba la condición **S AND Q = Q**
    - Ejemplo: empleadas del dpto. informática que sólo sepan castellano

      Q = 0001000000 0001 10000 10

- **Máscaras con bits que admitan cualquier valor:**
  - Se realizan con un bit para cada posible valor, en el conjunto { 0 , 1 , q }
  - La selección comprueba en **lógica trivaluada** la condición **S XOR - Q = 1**
    - Ejemplo: miembros del departamento de informática que sólo sepan inglés

      Q = qqq1qqqqqq qqqq 01000 qq

Tema 7.4.4: Estructuras Especiales

Esq. de bits Simples vs. Multiclave

- Es conveniente que el diseño de los esquemas de bits se realice atendiendo a las necesidades de procesamiento.
  - **Ejemplo** para claves A (5 bits) y B (10 bits) en un fichero de 1000 registros, con puntero de 3 bytes, y tamaño de bloque 2KB:
    - Tamaño índices:

T(A)=1000·(1+3) → 2 bloques T(B)=1000·(2+3) → 3 bloques	*Solución 1*
T(A+B)=1000·(2+3) → 3 bloques	*Solución 2*

→ caso i) Frecuencias relativas de uso: f(A)=0.7; f(B)=0.2; f(A+B)=0.1

- coste medio (solución 1) = 0.7*2 + 0.2*3 + 0.1*(2+3) = **2.5 accesos**
- coste medio (solución 2) = 0.7*3 + 0.2*3 + 0.1*3 = **3** accesos

→ caso ii) Frecuencias relativas de uso: f(A)=0.1; f(B)=0.4; f(A+B)=0.5

- coste medio (solución 1) = 0.1*2 + 0.4*3 + 0.5*(2+3) = **3.9** accesos
- coste medio (solución 2) = 0.1*3 + 0.4*3 + 0.5*3 = **3 accesos**

Tema 7.5.1: Procesos sobre Índices

Los índices son herramientas auxiliares versátiles que pueden ser utilizados de distintas formas con diversos propósitos.
Los usos más habituales son (1/2):

- ACCESO SIMPLE O UNÍVOCO (*index unique scan*): acceso que devuelve una entrada (o ninguna), con uno o más punteros.
  Ejemplos: - Index(DNI) → … where DNI = 1234567;
  - Index(nombre, apellido) → … where nombre = 'John' and apellido='Smith';

- ACCESO EN RANGO (*index range scan*): devuelve varias entradas en un rango ordenado. Según la naturaleza del índice, puede encontrarlas en un subárbol de un árbol B, en un segmento del encadenamiento de las hojas de un B+, o en búsqueda dicotómica extendida de un índice ordenado.
  Ejemplos: - Index(nombre, apellido) → … where nombre = 'John';
  - Index(fecha_ini) → … where fecha_ini between A and B ;

Tema 7.5.1: Procesos sobre Índices

Usos más habituales de un índice (2/2):

- ACCESO ALTERNO (*index skip scan*): acceso que utiliza un subconjunto de la clave de indización que no es prefijo.
  Ejemplo: - Index(nombre, apellido) → … where apellido = 'Smith';

- ACCESO A LA TOTALIDAD ESTRUCTURADO (*index full scan*): acceso que sigue la estructura de índice y recupera todas las entradas en orden.
  Ejemplo: - Index(nombre, apellido) → … order by nombre,apellido ;

- ACCESO A LA TOTALIDAD NO ESTRUCTURADO (*index fast full scan*): lee todos los nodos del índice aprovechando la secuencialidad de disco.
  Ejemplo: - Index(nombre, apellido) → select nombre,apellido from … ;

Tema 7.5.1: Proc. Ficheros Indizados

- Recuperación de **un registro** por clave identificativa (*exact match*):
  - 1 resultado, sin índice: coste según organización base
  - 0 resultados, sin índice: frecuentemente, coste más elevado
  - con índice arbóreo: pocos accesos índice + lectura de un cubo
- Comprobación de ***Unicidad*** (PK/UK) y ***Existencia*** (integridad FK):
  - Equivalente a la localización por clave identificativa
  - Índice suele ser ventajoso (en algunos SGBD, es obligado)
- Recuperación de **varios registros**:
  - Si *tasa de actividad* baja, coste(leer índice + *k* cubos) < coste(full scan)
  - Si *tasa de actividad* elevada, puede interesar ignorar el índice
  - Si proceso ordenado e índice ordenado, el ahorro de coste ordenación puede compensar la diferencia con el coste del full scan. Además, el índice proporciona resultados paulatinamente (no todo de una vez).

Tema 7.5.1: Procesos Indizados

• **Ejemplos**: fichero serial con 10^6 registros en 10^5 cubos; tenemos dos proc: buscar por K_1 (P_1) devuelve 1 reg, y buscar por K_2 (P_2) devuelve $2 \cdot 10^4$ (media)

- Para P_1 ¿es preferible leer 1 cubo (+acc índice) o hacer un full-scan?
- Si el soporte tiene beneficios secuenciales (disco), y el fichero de datos está desfragmentado, ¿compensa leer $2 \cdot 10^4$ cubos aleatorios o el full scan?
- Si el soporte no tiene ese tipo de beneficios ¿qué opción es preferible?
- ¿Y si en lugar de $2 \cdot 10^4$ fueran $2 \cdot 10^5$ registros? (cardinalidad(K_2)=5)
- Si la PK es K_1, y necesito insertar ¿miro que el nuevo valor para K_1 esté en el índice, o es preferible hacer un recorrido total (*full scan*)?
- Si K_3 es FK que referencia a una UK (no PK) e inserto un registro, para comprobar la integridad ¿hago un full scan del fichero referenciado?
- Si la búsqueda involucra varias claves privilegiadas ¿cuál escojo?

Tema 7.5.2: Localización Física

Filtrado de Cubos (*cjto dir. relevantes*)

¿ En qué cajón ha puesto mi hijo el móvil ?

A estos no llega…

De éstos no tiene la llave …

Puedo restringir la búsqueda a este subconjunto…

- Las *claves privilegiadas* permiten filtrar
- Los cubos que queden se recorren todos, a menos que se encuentre lo que se busca (*sel. identificativa*, en media (N+1)/2 accesos)

Tema 7.5.2: Filtrado por Índice

- El conjunto de direcciones relevantes puede ser un conjunto de punteros, o un simple vector de booleanos (uno por cada cubo en el fichero datos)

- Al seleccionar, se hará un **filtrado** gracias a la aplicación de índices.
  - una dispersión (hash) también puede filtrar sin coste adicional.
  - un índice primario proporciona un puntero → filtrado máximo
  - un índice secundario suele filtrar menos (puedo aplicar varios)

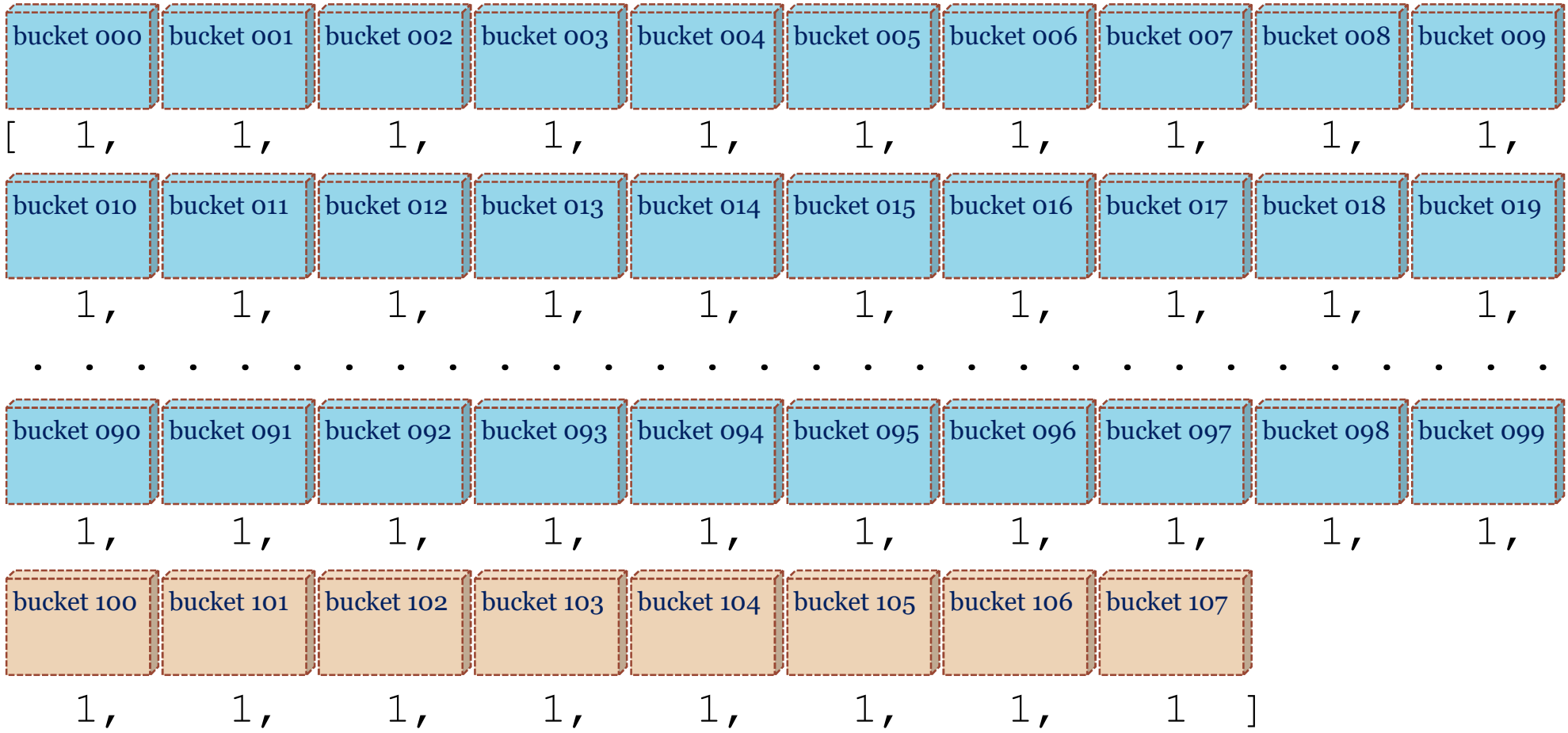
- La *selección indizada multiclave* consiste en aplicar varios filtros mediante otros tantos índices; se obtienen varios conjuntos que se operan (algebraicamente) para obtener el conjunto resultado global:
  - Ejemplos: sean *a* y *b* consultas simples; A y B sus respectivos cjtos. resultado

$$a \wedge b \equiv A \cap B$$
$$a \vee b \equiv A \cup B$$
$$a \wedge \text{-}b \equiv A - B$$
$$\text{-}\, a \equiv \aleph - A$$

(donde $\aleph$ es el espacio de direcciones global)

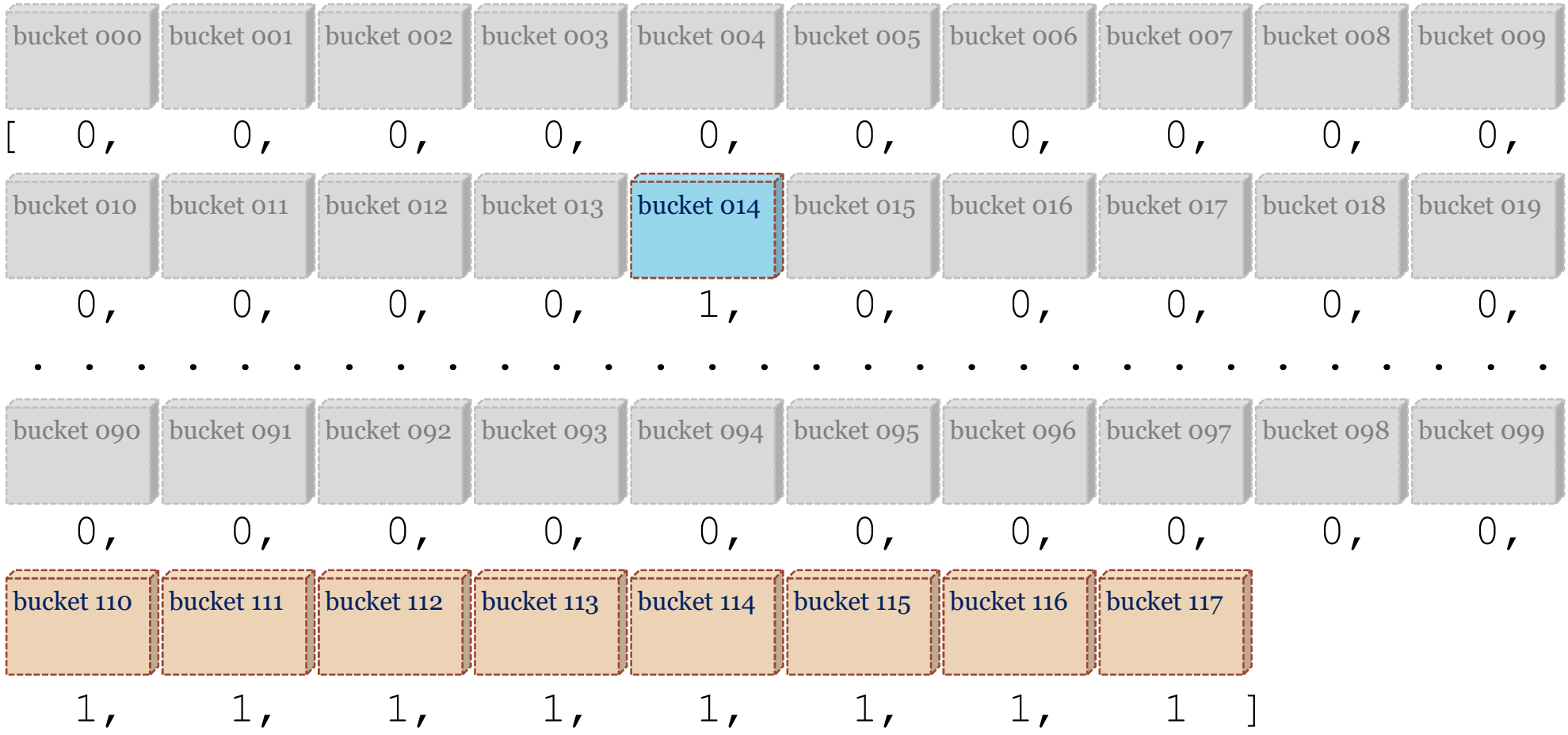
Tema 7.5.2: Ejemplo cjto. dir. relevantes

- Ejemplo: archivo disperso sobre N=100 con área desbordamiento N'=8



Tema 7.5.2: Ejemplo cjto. dir. relevantes

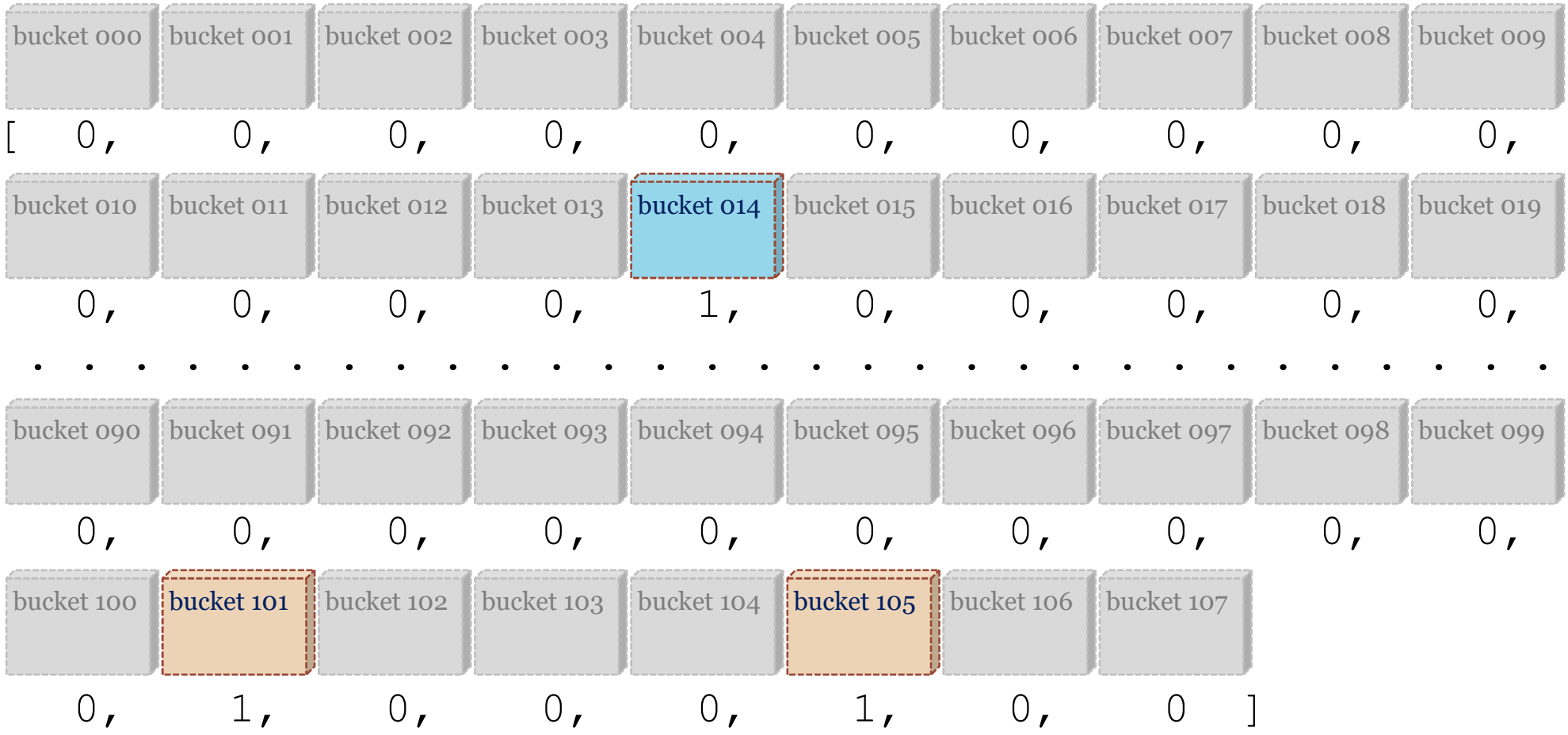
1. Filtra por CD='Juan', que produce la dirección 14



- …y descarta 99 cubos ! (el área desbordamiento no se descarta)

Tema 7.5.2: Ejemplo cjto. dir. relevantes

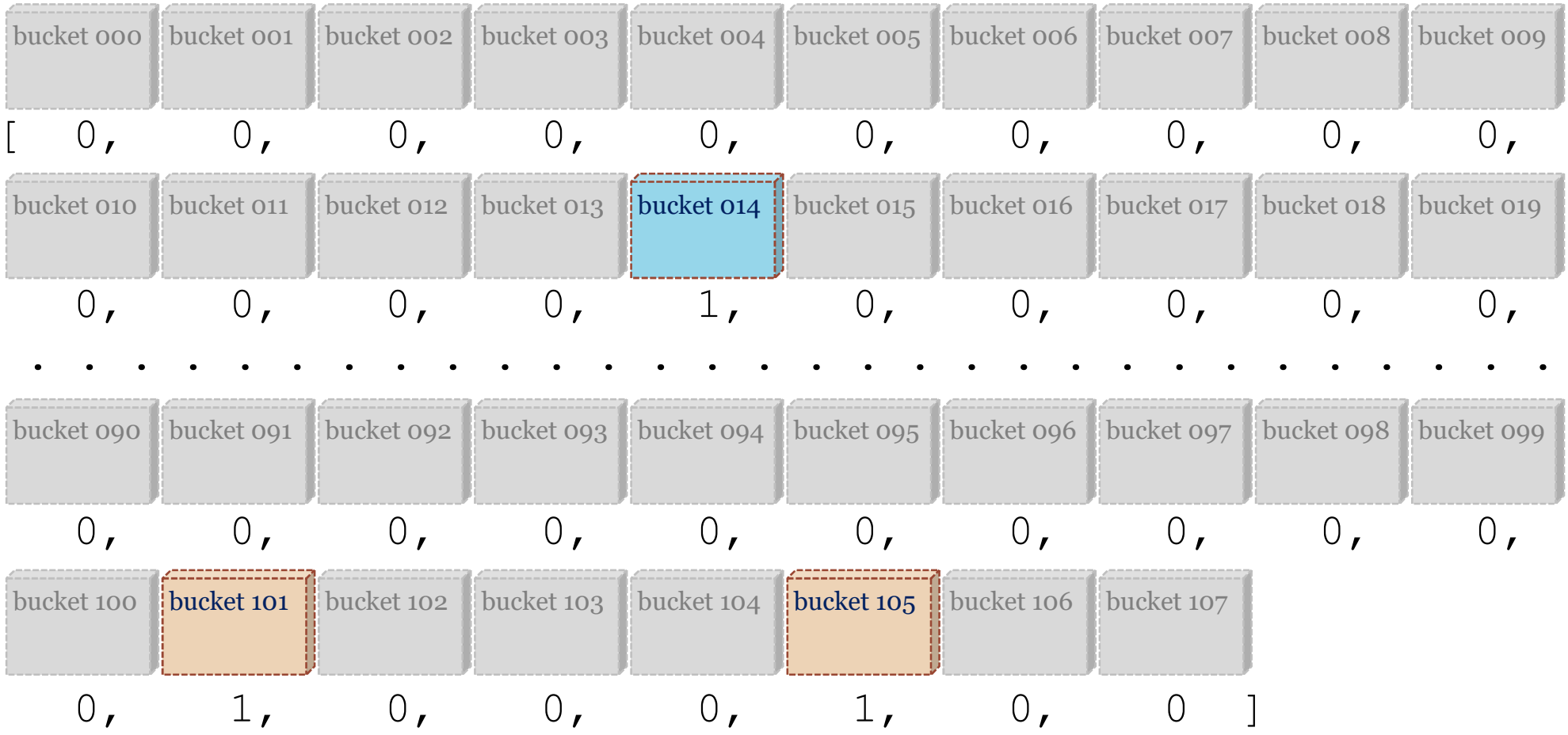
2. …y por índice *apellido*: 'Smith'·{3,9,14,21,28,44,56,62,89,94,101,105}



- Al ser 'conjunción', se calcula la intersección de cjtos. (op. más habitual)
- …descarta todo menos las 3 direcciones comunes

Tema 7.5.2: Ejemplo cjto. dir. relevantes

3. Recorre el archivo (*fullscan*), omitiendo cubos marcados con cero



- **Lee 3 cubos** (como mucho: si la selección fuera unívoca, se detendría en el primer resultado, realizando en media (3+1)/2 accesos de lectura)

Tema 7.5.3: Acceso Invertido

- El acceso invertido es un tipo de acceso indizado multiclave orientado a optimizar el coste de acceso en procesos muy concretos

- Se trata de procesos en los que se pretende averiguar información delimitada de ciertos archivos con condiciones muy concretas.

- Del tipo: *¿Cuál es el valor del campo X en el archivo Y, para los registros cuyo campo Z vale 'valor'?*

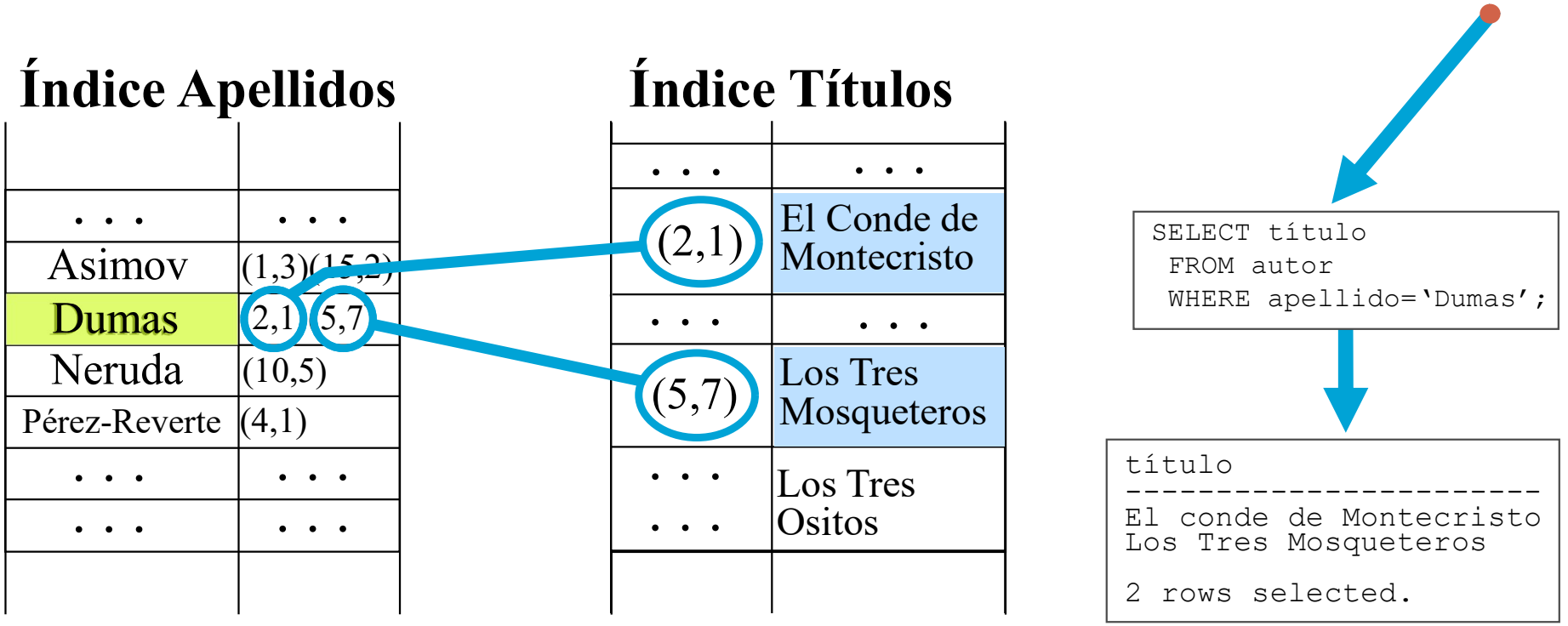
dos o más claves (pero no muchas)

- El acceso invertido procurará averiguar toda esta información **accediendo sólo a los índices** (sin acceder al archivo de datos)

- Sus punteros relativos deben localizar unívocamente cada registro:
  → punteros con parte alta (***cubo***) y parte baja (***posición*** *en el bloque*)
  → Oracle usa el ROWID, que contiene *dbblock* (cubo), *position*, y *datafile*

Tema 7.5.3: Acceso Invertido

- Se ejecutarán primero las condiciones (para obtener conjunto resultado)
- Después se busca en los 'índices objetivo' (incógnitas), ¡pero al revés!:
  a partir de cada dirección (ptro. relativo) buscamos el valor (ptro. lógico)
- **Ejemplo**:
  ¿Cuáles son los títulos de los libros de cualquier autor cuyo apellido sea 'Dumas'?



Tema 7.5.3: Acceso Invertido: Coste

- El coste del acceso invertido es la suma de los costes de sus dos partes:
  - Selección: *Obtención de los punteros*
    - *Listas invertidas no ordenadas: coste máx.* ***n****; medio (n+1)/2*
    - *Listas invertidas ordenadas:* $\boldsymbol{\log_2 (n+1)}$
    - *Esquemas de bits:* ***n***
    - *Otro tipo de índice: el coste correspondiente a esa estructura*
  - Proyección: *obtención de Claves correspondientes a los punteros*
    - *Esquemas de bits con puntero implícito:* ***mín (n, r)***
    - *Cualquier otro caso:* ***n***
- Simbología: ***n*** es el número de bloques del índice; ***r*** es el número de resultados
- Si hubiera varios índices implicados en la condición o en la proyección, el coste en cada parte sería la suma de los costes individuales de cada índice implicado.

Tema 7.5.3: Acceso Invertido

- El acceso invertido es eficiente si requiere acceder a **pocos índices** (si accediera a varios no contenidos en M_{int}, podría costar más que acceder a los datos).
- También es eficiente si la misma **consulta se repite** frecuentemente (accediendo a los mismos índices, se dispara la tasa de acierto a memoria intermedia).
- Los índices que soportan este acceso (puntero ext. doble precisión) se denominan **índices invertidos**. Los secundarios también se denominan listas invertidas.
- En este tipo de acceso, los índices bitmap son eficientes en dominios reducidos, mientras que las listas invertidas lo son en dominios de cardinalidad elevada.
- Un **fichero invertido** es el que soporta este tipo de acceso (dos o más índices invertidos). Un **fichero totalmente invertido** tiene todos los campos invertidos.
- En un fichero totalmente invertido el área de datos es redundante (prescindible). Sin embargo, se suele mantener para (a) evitar el alto coste de la recomposición de registros; y (b) mantener doble almacenamiento base (**sistema dual**).

Tema 7.5.3: Sistema Dual

- Un **sistema dual** es el que presenta redundancia controlada para disfrutar de alternativas de acceso físico (dependiendo del tipo de consulta, se utiliza un recurso u otro). Por ejemplo, consultas de pocos atributos por acceso invertido, y el resto sobre el área de datos.
- La duplicidad de almacenamiento en sistemas duales pone en riesgo la **consistencia**. Controlar la redundancia es costoso, porque las actualizaciones deben realizarse en varios almacenes de modo atómico.
- Además, también puede afectar a la **disponibilidad** del sistema, ya que el control del proceso debe esperar a recibir confirmación de todas las escrituras.
- Para mejorar la disponibilidad se puede sacrificar algo de consistencia: uno de los almacenes se actualiza efectivamente (al realizar la operación) y el otro eventualmente (por ejemplo, los índices se actualizan cuando el sistema está ocioso), mejorando la disponibilidad con poco perjuicio en las consultas.

Tema 7.5.3: Fusión de Índices (*Index Join*)

- En ficheros invertidos, es habitual combinar varios índices a través de su identificador de registro (puntero o ROWID) hasta tener un índice que contiene todos los atributos involucrados en la consulta. A este proceso se le conoce como fusión de índices (index join).

- El coste en accesos de la fusión (completa) de índices es el de recorrer a la totalidad (full scan) los índices involucrados en el proceso.

- La fusión de índices es eficiente si se hace sobre índices en mapa de bits. Algunos SGBD (como Oracle) contemplan convertir otros índices secundarios (árbol B+) en bitmap en memoria para realizar este proceso.

- El coste de la *index join* es proporcional a la cantidad de índices involucrados. Recrear el área de datos (recomposición total) es un proceso costoso, y si se realiza de modo frecuente es preferible contar con un sistema dual.