Tema 6: Acceso Multi-Clave

- **Introducción: problemática del acceso multiclave**
- **Acceso Invertido**
  - *Listas invertidas*
  - *Esquemas de Bits*
  - *Sistemas Duales*
- **Selección Multiclave: accesos básico e indizado**
- **Organizaciones Indizadas para acceso multiclave**
  - *Árboles* ***R***
- **Consultas *n-dimensionales***
- **Organizaciones Direccionadas Multiclave**

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6: Introducción

El acceso selectivo tiene dos vertientes:

- simple: se busca un registro (o varios) a través de una clave
- multiclave: el proceso de selección afecta a varias claves
  - el criterio de búsqueda contempla varias claves
  - el resultado no implica el registro completo, sino una clave

- El acceso multiclave suele basarse en claves no identificativas
- Se puede aplicar sobre organizaciones clásicas, si bien existen algunas organizaciones específicas que pueden reducir el coste
- El acceso multiclave (o *n-dimensional*) es de gran interés por sus posibilidades y aplicaciones, como las BBDD espacio-temporales

© 2008 LaBDa – Universidad Carlos III Madrid

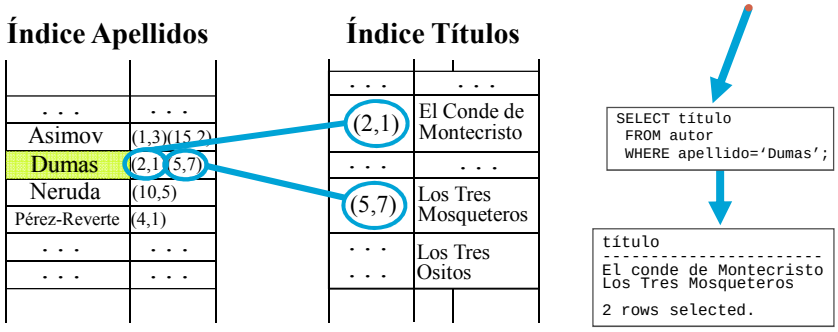
Tema 6.1: **Acceso Invertido**

- 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'?
- 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 (***bloque***) y parte baja (***posición*** *en el bloque*)

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1: 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 libros del autor cuyo apellido es 'Dumas'?*



© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.1: Inversión por Listas

- El acceso invertido es eficiente si requiere acceder a pocos índices (si accediera a varios no contenidos en M_{int}, costaría más que acceder a los datos)
- Los índices involucrados pueden ser primarios o secundarios
- Puede utilizarse cualquier organización indizada (invertida, con punteros completos), pero también existen estructuras específicas.

Listas Invertidas

Una lista invertida para un campo de un fichero consiste en una lista ordenada de todos los posibles valores de ese campo, a los que se asocian referencias a todos los registros que llevan ese valor.

Ejemplo:

```
Asimov.............. (01,05),(17,02)
Dumas............... (02,03),(11,06)
Neruda.............. (10,01),(13,04)
Pérez-Reverte....... (04,03),(06,02),(09,01)
```

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.1: Listas Invertidas Características

- Son entradas de índice (*'como un índice secundario agrupado por valores'*) (o bien 'como un índice que en lugar de uno tiene varios punteros relativos)
- Se trata de un índice ordenado (por la clave)
- Cada puntero consta de partes *alta* (dir. bloque) y *baja* (pos. en el bloque)
- Esas entradas de índice tienen tamaño variable (de clave y de lista de punteros) *(posible implementación: long_clave + clave + long_lista + lista_punteros)*
- Los valores se repiten en general:
  - un valor en varios registros → **clave secundaria**
  - varios valores para un registro → **campo multivaluado** (grupo repetitivo)

© 2008 LaBDa – Universidad Carlos III Madrid

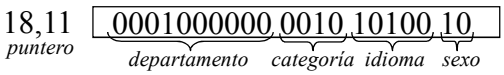
Tema 6.1.2: Esquemas de bits

Un esquema de bits para un campo es un vector de valores booleanos, de modo que a cada posible valor se le hace corresponder una posición

Ejemplo: idioma (castellano, inglés, francés, alemán, italiano)
el esquema de idioma para Juan Valdés: (10000)

- Si el número de valores posibles de un campo fuera pequeño, se podría mantener un índice que en lugar de entradas [valor, puntero] tuviera [puntero, esquema de bits].

- Se pueden considerar esquemas de varios campos concatenados



© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.2: Usando Esquemas de bits

- Los esquemas de bits pueden hacer referencia a uno o varios campos (fichero parcialmente invertido) o incluso a todos (totalmente invertido), indizando algunos valores (i. parcial) o todo el dominio (i. exhaustivo).

Ejemplo:

apuntamiento	*departamento categ. idioma sexo*
18, 11	0001000000 0010 10100 10
10, 8	0001000100 0100 10000 10
12, 5	0001000000 0010 11100 01
. . .	. . .

- Estos esquemas hacen especialmente eficiente el acceso multiclave
- Además posibilitan indizar varios valores en la misma entrada (para campos multivaluados), mejorando el acceso basado en varios valores.
- La selección se hará comparando una *máscara* (*query*) con cada uno de los esquemas de bits almacenados, devolviendo los punteros que encajan

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.2: Esquemas de bits
Simples vs. Combinados

- 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**

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.2: Esquemas de bits
Puntero Implícito

- Las entradas por esquemas de bits suelen incluir un puntero (relativo) que apunta al registro al que hace referencia el registro.
- Este se puede omitir si se puede averiguar la posición del registro a partir de la posición del esquema de bits (puntero implícito).
- Para ello, es necesario que la organización base esté formada por cubos de tamaño fijo (o, al menos, acotados superiormente).
- **Ejemplo**: el archivo de la diapositiva anterior presenta 300 cubos de tamaño máximo 5. El esquema de bits necesitaría tener 'reservados' 300*5=1500 esquemas de bits (aunque sólo 'ocurran' 1000).
  - para la clave A (5 valores) → 1500*1 byte → 1 bloque
  - Para la clave B (10 valores) → 1500*2 bytes → 2 bloques
  - Para la clave A+B (15 valores) → 2 bloques

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.2: Máscaras y Esquemas de bits

- **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: gente del departamento de informática que sólo sepa inglés

      Q = qqq1qqqqqq qqqq 01000 qq

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.3: 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.

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.4: Comparativa (I)

Listas Invertidas vs. Bitmaps

- El acceso por **esquema de bits** es muy eficiente
  - las entradas tienen tamaño fijo
  - operaciones de actualización son inmediatas (muy eficientes)
  - comprobación de la entrada por aplicación de máscaras (eficiente)
  - admite **multivaluación** e *índices multiclave*
- Pero queda limitado a:
  - claves cuyo dominio (posibles valores) sea discreto y reducido (para evitar esquemas de bits demasiado largos)
  - condiciones sencillas (igualdad/desigualdad)
  - ficheros de volumen medio-bajo (no demasiados registros) (¡procesa todas las entradas serialmente!)

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.4: Comparativa (II)

Listas Invertidas vs. Bitmaps

- El acceso por **listas invertidas** es complementario:
- es más eficiente cuanto mayor es el dominio (número de posibles valores)
  - → a más valores más listas
  - → cada lista tendrá menos punteros (listas invertidas más pequeñas).
  - → al ser más pequeñas, son más manejables (por ejemplo, si se implementan en árbol B, mayor será el orden del árbol)
- Además:
  - buen comportamiento con volúmenes grandes (número elevado de registros)
  - admite *multivaluación*, pero no *multiclave* (a menos que se fusione con *R-tree*)
  - las condiciones que pueden aplicarse son algo más complejas (p.e., rangos)
- Sin embargo, el mantenimiento y manejo son muy costosos:
  - actualizaciones costosas (desplazamiento de entradas, ...) → *no consecutividad*
  - manejo de mucha información (en Mppal y secundaria)
  - algunas condiciones de selección siguen siendo muy costosas.

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.5: Fichero Invertido

Se dice de un **archivo** (fichero) que está **invertido** si tiene índices invertidos para todos sus campos (*totalmente invertido*) o bien solamente para algunos de ellos (*parcialmente invertido*)

- Un fichero totalmente invertido es **redundante**. Se podría decidir no almacenar los campos invertidos (que cuentan con una lista invertida) y después **ensamblar** esa información cuando sea necesario.
- Caso extremo: un fichero totalmente invertido podría minimizar el almacenamiento en el fichero maestro (hasta desaparecer)
- Este ahorro de espacio supone un coste muy elevado en tiempo si existe algún proceso de recuperación del registro completo.
- Admite optimizaciones dependiendo (del propósito) del sistema

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.1.6: Sistemas Duales

- El acceso serial tiene la ventaja de no precisar almacenamiento auxiliar, fácil inserción, y además se puede aplicar cualquier condición (compleja). Por otro lado, el tiempo de localización es malo (peor cuanto más grande).
- Por todo ello, es habitual proponer **Sistemas Duales**:

  Doble organización física de los datos (redundancia controlada)

  - Se cuenta con una organización base consecutiva, que simplifica las actualizaciones y que posibilita selecciones complejas.
  - Se incorpora una organización invertida (adecuada a las necesidades) para facilitar el acceso a través de determinadas claves (y valores).
  - ¡Una de las organizaciones (la invertida) no se mantiene actualizada! (se actualizan periódicamente: noches, fines de semana, cuando el sistema está ocioso ...)
  - Pueden no invertirse todos los valores (ni todos los registros de un valor...)

  → Esto **obliga a repetir muchas consultas (infructuosas)**

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.2: Selección Multiclave

***Acceso Multiclave Básico*: Búsqueda Serial**

- Consiste en recorrer todos los registros del fichero de modo consecutivo comprobando cuáles cumplen la condición de búsqueda.
- *Ventajas*
  - por compleja que sea la condición, el coste (acceso) no se ve afectado.
  - no precisa almacenamiento auxiliar
- *Inconvenientes*
  - Es una búsqueda serial (*full scan*): recorre todo el área de datos
  - el **coste es muy elevado** (n) y se hace inmanejable en ficheros grandes
- Las mejoras basadas en ordenación no cubren todas las combinaciones.
  Ejemplo: nombre,ap1,ap2; en ausencia del nombre, no se aprovecha la ordenación

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.2: Selección Multiclave

Indización Multi-Clave

- La indización multi-clave sólo se aplica sobre índices secundarios
  (si involucrara un primario, el resultado de este sería una dirección → un acceso)
  - **Consultas multiclave** para claves soportadas por un índice secundario:
    - se descompone la consulta en subconsultas sobre un sólo indice cada una; cada subconsulta aportará un conjunto resultado (parcial)
    - se operan los conjuntos resultado mediante álgebra de conjuntos para reconstruir la consulta original
    - se obtiene un conjunto de direcciones candidatas (donde existen encajes)
  - Ejemplos: sean *a* y *b* consultas simples; A y B sus respectivos cjtos. resultado

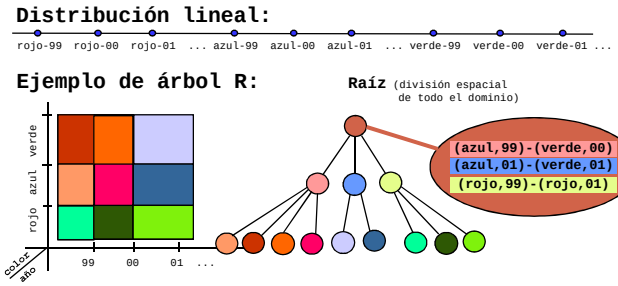
$$a \wedge b \equiv A \cap B$$
$$a \vee b \equiv A \cup B$$
$$a \wedge \text{-}b \equiv A - B$$
$$\text{-}a \equiv \aleph - A$$

(donde $\aleph$ es el total de las direcciones de cubos ocupados)

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.2.1: Indización Multi-Clave
El árbol **R** (**R-tree**)

- 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 (propuesto por Güttman en 1984)
- Cada entrada no es un punto en una línea, sino un intervalo d-dimensional:



- 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)

© 2008 LaBDa – Universidad Carlos III Madrid

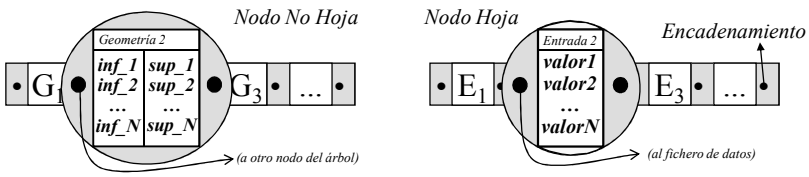
Tema 6.3.1: Árboles **R** (**R-trees**)
Aspecto de los **nodos**

Uso: especialmente útiles para el manejo de información espacio/temporal

Preliminar: todo dominio presenta una relación de orden

Entradas: están todas en los nodos hoja (resto de nodos, para localizar hojas)

- **HOJAS**: posición n-dimensional y su correspondiente puntero externo
  `(valor1, valor2, ..., valorN, puntero_externo)`
- **RESTO**: geometría n-dimensional y su correspondiente puntero interno
  `(rango1, rango2, ..., rangoN, puntero_interno)`

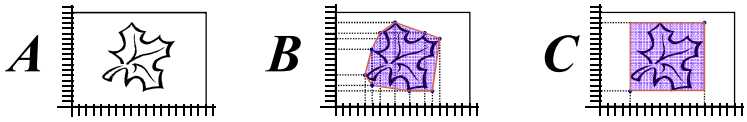


© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.2.1: Árboles R (R-trees)

Almacenando Geometrías

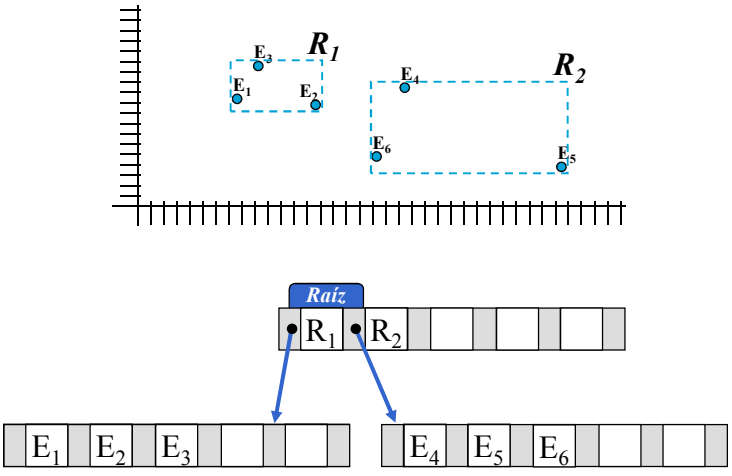
- Todos los punteros externos están en nodos hoja, con sus entradas
- Las entradas pueden ser puntos, **¡pero también geometrías!** (fig. A)
  La hoja no ocupa un punto, sino un área (definida por una geometría)
- Si la geometría es muy compleja, ocupará más espacio (en la fig.B, implica almacenar ¡9 puntos!) → conviene utilizar **Geometrías Regulares**
- El ***intervalo mínimo multidimensional*** **(Minimun Bounding Box - MBB)** consiste en una geometría regular definida por dos puntos (fig. C):
  inferior (en todas sus dimensiones) y *superior* (en todas sus dimensiones).



© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.2: Árboles R (R-trees)

Almacenando Entradas



© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.2: Árboles **R** (**R-trees**)

Políticas de **Partición** (*Splitting Policies*) I

- ***Partición***: a partir de un área R_i que contiene un conjunto S_i de elementos (geometrías o puntos n-dimensionales) obtener otras dos (o más) áreas R_j y R_k que contengan todos los elementos de S_i.
  - Aunque no es obligatorio, se suelen buscar dos áreas R_j y R_k
  - Esas áreas serán las MBB que contengan los subconjuntos S_j y S_k
  - La política de partición deberá decidir el *reparto* de todos los elementos de Si en los dos subconjuntos, tal que $S_j \cup S_k \equiv S_i$
- La **política de partición** será la que decida el **reparto**:
  - en cuantos subconjuntos (generalmente dos);
  - si esos subconjuntos están o no equilibrados (reparto de carga);
  - que criterio se escoge para dividir un conjunto en varios (criterio)

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.2: Árboles **R** (**R-trees**)

Políticas de **Partición** (*Splitting Policies*) y II

- **Reparto**: aunque no es obligatorio, se tenderá en general al equilibrio de carga entre ambos subconjuntos
  → páginas descendientes llenas al **50%**

Criterios de reparto:

- **Agrupamiento**: para cada subconjunto candidato S_c se hallará el centro de gravedad (c.d.g.);
  se buscará el reparto que minimice el sumatorio de las distancias de cada elemento al centro de gravedad de su conjunto.
- **MBBs mínimos**: se procura minimizar el producto de los intervalos de las MBB (su área, su volumen,...)
- **MBBs regulares**: se procurará partir por la dimensión cuyo rango en R_i sea (relativamente) mayor

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.3: Árboles R (R-trees)

Propiedades (I)

Ocupación Máxima y Mínima:

- Cada nodo tiene un máximo de entradas (***k***) y de descendientes (***m***)
  - → se calculan por el tamaño de página, y de ellos se saca la ocupación mínima
    - Todas las hojas (no raíz) tienen un mínimo de entradas (k_{min})
    - La raíz (no hoja) tiene un mínimo de dos descendientes
    - Los nodos internos tienen un mínimo de descendientes (m_{min})
    - Los nodos no hoja tienen tantas entradas como descendientes
    - En general, **m = k** (aunque existe una variante con $m = k + 1$)

· ***Cálculo del Orden*** (***m***): si el reparto es equilibrado en dos descendientes...

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

tal que... $T_{entrada} = 2 \cdot (\Sigma\, T_{clave\ i}) + T_{ptro\_externo}$

y además...

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

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.3: Árboles R (R-trees)

Propiedades (II)

Profundidad del árbol:

- El árbol R es balanceado en altura (crece de abajo-arriba)
- Se puede acotar superiormente el número de hojas:

  Nº máximo de hojas = nº de entradas / ocupación mínima →

$$n_{hojas} = \left\lfloor \frac{k_{min}+1}{2} \right\rfloor$$

- También se puede acotar superiormente el número de ascendientes en el *nivel n-1* (la fórmula para hallar esa cota es la misma)
- Aplicando iterativamente esa fórmula hasta llegar a la raíz (nivel cuyo número de nodos sea 1) se podrá calcular la profundidad del árbol (***n***)
- La profundidad del árbol determina el número mínimo de accesos.
- Si los intervalos no se solapan, también determina el coste máximo de algunos tipos de acceso (concretamente, de *exact match* y *point query*)

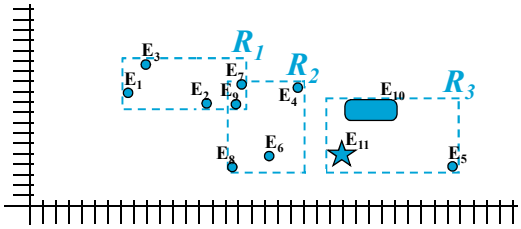
© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.4: Árboles R (R-trees)

Operaciones de Actualización

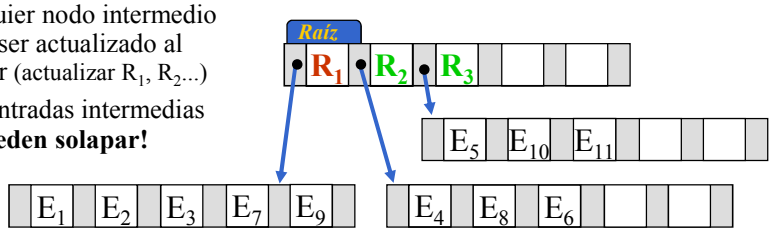
Inserción:

- En su MBB
- Si entra en varios, el menor
- Si no entra en ninguno, el que vaya a crecer menos con la nueva inserción…



Corolarios:

- Cualquier nodo intermedio puede ser actualizado al insertar (actualizar R_1, R_2...)
- Esas entradas intermedias **¡se pueden solapar!**



© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.4: Árboles R (R-trees)

Operaciones de Actualización

Borrados:

- **Condensación del árbol**: por efecto de borrados, la ocupación de un nodo puede bajar del umbral establecido. En tal caso se eliminará ese nodo, reubicando las entradas que contenía.
- **Ajuste** de MBB's: en ocasiones, al borrar un elemento, el MBB ascendiente puede reducirse sin dejar de contener el resto de entradas

Modificaciones:

- Si no afecta al MBB ascendiente, se actualiza la entrada en su nodo
- En caso de afectarlo, lo mejor es eliminar la entrada y reinsertarla

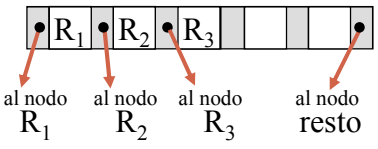
© 2008 LaBDa – Universidad Carlos III Madrid

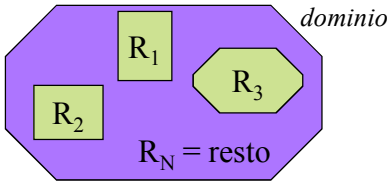
Tema 6.3.5: Árboles **R** (**R-trees**)

Variantes del Árbol-**R**

Variante del diseño de un nodo (variante de página):

- En todo nodo no hoja, se puede tener una entrada *'resto'*.
- Dicha entrada tendría la geometría definida por el MBB ascendiente menos las geometrías definidas en el nodo. Por no ser una geometría explícita, no se almacenaría en el nodo: sólo se almacena su puntero interno (como en los B, el nodo tendría un puntero más que entradas).
- Si es la raíz, el resto es el dominio n-dimensional menos las geometrías que contiene explícitamente.





© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.5: Árboles **R** (**R-trees**)

Variantes del Árbol-**R**

- **Árbol R empaquetado** (Packed R-tree): partición óptima del universo y árbol R mínimo (óptimo). Precisa conocer los elementos a priori.
- **Árbol R esférico** (Sphere R-tree): esferas n-dimensionales en lugar de MBB
- **Árbol P** (P-tree): utiliza poliedros (definidos o predefinidos) en lugar de MBB

- **Árbol R*** (R* tree): aumenta la densidad y reduce la superposición de MBB
  - **Forzar reinserción**: cuando un nodo desborda, en lugar de dividirlo se toma cierta cantidad de sus entradas (por poner algunos ejemplos: 1, 2, ó el 30%) y se reinsertan (escogiéndolas bien, a menudo se evita la partición del nodo).
  - Esa técnica también se utiliza cuando se superponen dos MBB del mismo nodo, de modo que se minimiza la superposición de intervalos en un nivel.
  - Al extender MBB, se procura evitar superposición y reducir su perímetro.

Resultado: árboles más densos y aplastados, y búsquedas de camino único.

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.3.5: Árboles R* (R*-trees)

Forzando la Re-Inserción

¡¡¡ R1 DESBORDA !!!

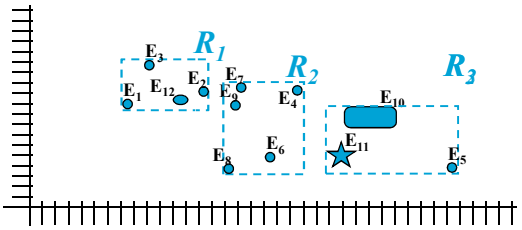
Escogeremos una (o más) entrada(s) para reinsertar

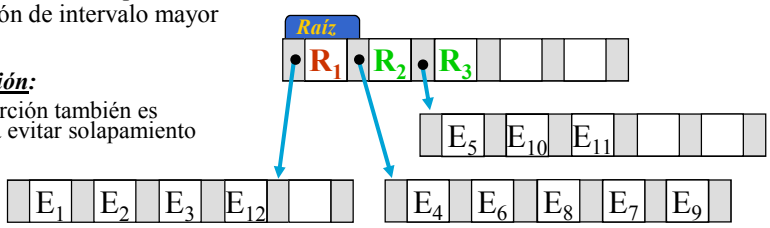
1.- Criterios de Selección:

- que pertenecen a otro rango
- que está más alejada del *cdg*
- que reduce el MBB
- que reduce el MBB por la dimensión de intervalo mayor

Observación:

- la reinserción también es útil para evitar solapamiento



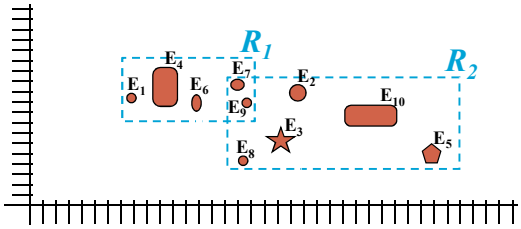


© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.4: Consultas *n-dimensionales*

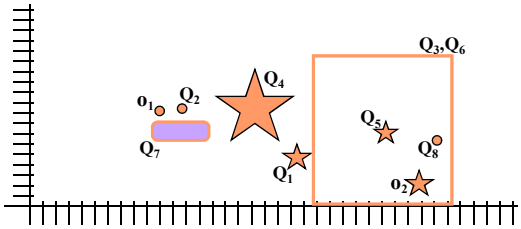
Tipos de Consulta:

1. Exact Match Query (**EMQ**)
2. Point Query (**PQ**)
3. Window Query (**WQ**)
4. Intersection Query (**IQ**)
5. Enclosure Query (**EQ**)
6. Adjacency Query (**AQ**)
7. Containment Query (**CQ**)
8. Nearest-Neighbor Query (**NNQ**)
9. Spatial Join (**SJ**)



Ejemplos:

- $EMQ(Q_1) = \{E_3\}$
- $PQ(Q_2) = \{E_4\}$
- $WQ(Q_3) = \{E_2,E_5,E_{10}\}$
- $IQ(Q_4) = \{E_7,E_9\}$
- $EQ(Q_5) = \{E_{10}\}$
- $AQ(Q_6) = \{E_{10}\}$
- $CQ(Q_7) = \{E_4,E_6\}$
- $NNQ(Q_8) = \{E_5,E_{10}\}$
- $SJ(\{E_i\}\cdot\{o_i\}, \text{`distancia}(Q_i,E_i) < 3\text{'}) = \{(o_1,E_1),(o_1,E_4),(o_2,E_5)\}$



© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.4: Consultas *n-dimensionales*

Ejemplos Prácticos

¿Qué tipo de consulta responde a las siguientes preguntas?

1. Coche aparcado en la plaza 555
2. ¿En qué edificio está el vigilante?
3. Jugadores en el campo de fútbol (en un intervalo 2D y de tiempo)
4. Carreteras que pasan por Leganés
5. Ámbitos geográficos del Bierzo
6. Provincias limítrofes con Segovia
7. Contar los policías que se encuentran en la zona centro
8. Ambulancia más cercana al lugar de un accidente
9. Ciudades atravesadas por ríos

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.4: Consultas *n-dim* con **R-trees**

Ejemplos de Implementación en (Spatial)Oracle

Window Query:

```
SELECT A.feature FROM TABLA A, VENTANAS B
       WHERE  B.activa='S' AND
              sdo_filter(A.shape, B.shape)='TRUE';
```

Nearest Neighbour-Query:

```
SELECT H.feature FROM HOSPITALES H, AMBULANCIAS A
       WHERE  A.matrícula='0000AAA' AND
              sdo_nn(A.shape, H.shape,NULL)='TRUE';
```

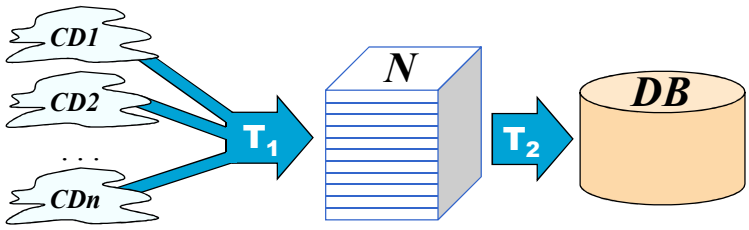
Spatial Join:

```
SELECT C.nombre, R.nombre
       FROM   TABLE(SDO_JOIN('CIUDADES','SHAPE',
              'RIOS','SHAPE','mask=FILTER') J,
              CIUDADES C, RIOS R
       WHERE  J.rowid1=C.rowid AND J.rowid2=R.rowid;
```

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.5: El Direccionamiento Disperso Multi-Clave

El Acceso multiclave sobre organización direccionada consiste en ampliar el algoritmo de transformación para dispersar combinaciones (no unívocas) de valores de varias claves sobre cubos.



- Observar la diferencia entre un direccionamiento cuya CD sea $a+b+c$ y un direccionamiento multi-clave con $CD_1 = a$, $CD_2 = b$, y $CD_3 = c$
  - ☞ *en el primer caso es necesario aportar tres valores para obtener beneficios*

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.5.1: Dispersión Multiclave

Cuando la dispersión es estática (espacio de direccionamiento fijo) el algoritmo de transformación multiclave sigue los siguientes pasos:

(T_1)
- transformación de cada clave en un componente de la dirección, siguiendo las funciones de transformación clásicas.
- combinación de todos los componentes obtenidos (en una o más direcciones base). La combinación básica es la concatenación.

Ejemplo:

CD_1 = 'Calle' → 3+1+12+12+5 MOD 32 → 11111
CD_2 = 'Gómez' → 7+16+13+5+27 MOD 32 → 00100
CD_3 = ? → ? → xxx

`1111100100XXX`
(8 bloques relevantes)

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.5.1: Dispersión Multiclave

Es necesario escoger bien los subespacios: las claves más frecuentemente utilizadas en las búsquedas han de tener más espacio para dispersarse, mientras que a las claves infrecuentes puede asignársele un subespacio menor (optimizando así el tiempo medio de acceso):

Ejemplo: CD_1 = primer apellido, CD_2 = segundo apellido
Búsquedas con 1 clave: 90% CD_1 y 10% CD_2

- n=16 → $n_1 = 8$ y $n_2 = 8$
  con este reparto, buscar con 1 clave supone 256 bq relevantes
- n=16 → $n_1 = 10$ y $n_2 = 6$
  así buscar con 1 clave son $2^6 \cdot 0.9 + 2^{10} \cdot 0.1 = 160$ bq relevantes

Nota: *n* es el número de bits en la dirección binaria del subespacio
Si n=16, entonces $N = 2^{16} = 65536$, que es un espacio lineal gigantesco

© 2008 LaBDa – Universidad Carlos III Madrid

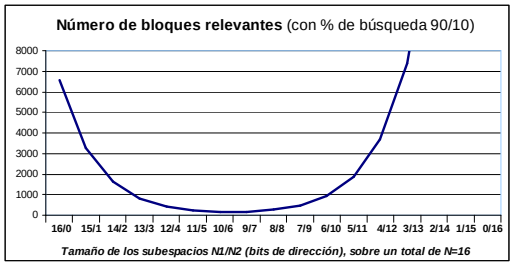
Tema 6.5.1: Dispersión Multiclave

Cálculo de Subespacios

- Si uno de los subespacios tiene $N_2 = 0$, equivale a un direccionamiento normal. (buscar por N_1 sería 1 acceso y por N_2 sería una búsqueda serial en todo N)
- El reparto óptimo depende del espacio completo (n) y de los porcentajes de uso (p_i)

* ***Sobre dos subespacios:***

$$n_i = \frac{\log_2 \frac{p_i}{100-p_i} + n}{2}$$



* ***Sobre más subespacios:***

- *hallar el primero (n_i),*
- *para el siguiente, actualícese la fórmula, sustituyendo: 100 → ($100-p_i$) ,, n → ($n-n_i$)*
- *y sucesivamente, se emplea ($100-\Sigma p_i$) y ($n-\Sigma n_i$) para los i subespacios ya calculados*

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.5.2: Disp. Virtual Multiclave (apoyada en Directorios)

La dispersión multiclave 'reparte' la velocidad del acceso direccionado entre varias claves. Hay que utilizarlo cuando la búsqueda por todas las claves (clave global) es probable, y también lo es cada clave individual.

- No deben incluirse claves poco utilizadas, porque aumenta el tamaño del área de datos y el nº de bloques relvs. en cada búsqueda que no incluya esa clave.
- El aumento excesivo del área de datos tiene como consecuencias que aparezcan bloques vacíos, que bajan la densidad y malgastan accesos.

• Para mejorar estas organizaciones se suele incorporar una estructura auxiliar (directorio), de tamaño reducido (bloqueada en $M_{intermedia}$) que señale los cubos vacíos (para no acceder a ellos), incluso que establezca una nueva dirección real para los ocupados (y no almacenar los vacíos).

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.5.2: Disp. Virtual Multiclave (apoyada en Directorios)

Tipos de directorio (dispersión virtual):

I. **Directorio de ocurrencias:** por cada cubo, un bit
Ese *bit* indica si el cubo está vacío (0) o contiene algo (1).
Se ahorran accesos al descontar los bloques relevantes vacíos.

II. **Directorio virtual:** por cada cubo, un puntero
Utiliza direcciones virtuales para compactar el fichero de datos.

La siguiente optimización consistiría en adaptar el área de datos a las necesidades puntuales del fichero → disp. virtual extensible multiclave

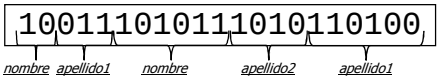
- En 1982, Tamminen propone esa disp. virtual extensible multiclave (**EXCELL**)
- En (1981-1984) Nievergelt, Hinterberger y Sevcik proponen los ***ficheros rejilla*** como un *'cuadriculado'* d-dimensional del universo.

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.5.3: Dispersiones Multiclave (Extensibles y Dinámicas)

• Para describir la dispersión extensible multiclave (así como la dinámica), hay que considerar extensiones para cada dimensión:
cada componente de la dirección tendrá más bits de los usados inicialmente

- Se ha de disponer de funciones de dispersión de muchos bits (m_i)
- Inicialmente, sólo se usan k bits (k<m), que definen $k_1 \cdot k_2 \cdot \ldots \cdot k_d$ clusters (cada cluster pertenece a una dimensión, y pueden repetirse...)



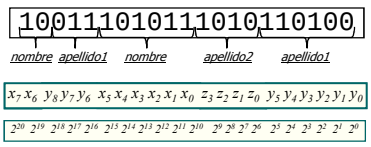
- Cuando un cluster se satura, se divide añadiendo l bits de la dimensión j (la que mejor disperse), tal que $k_j + l < m_1$ (en general, supondremos l=1)
- Esos bits se añaden en la parte alta (a la izquierda) de la dirección virtual (de modo que no hay que rehacer el directorio, solo replicarlo 2^l veces)

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.5.3: Dispersiones Multiclave (Extensibles y Dinámicas)

- Es deseable que el almacenamiento de los bloques sea lineal
- Para poder mantener una correspondencia entre direcciones y bloques, se hará uso de un directorio multidimensional, que irá asignando bloques vacíos a cada entrada del directorio (*d-dimensional*) que entre en uso.
- La localización en el directorio n-dimensional sigue la concatenación: la entrada $(v_1, v_2, \ldots, v_d)$ se halla en la posición $v_1 + 2^{k_1} \cdot (v_2 + 2^{k_2} \cdot (\ldots \cdot (v_d) \ldots))$

Ejemplo: *nombre* $\equiv x_9 \cdot x_8 \cdot x_7 \cdot x_6 \cdot x_5 \cdot x_4 \cdot x_3 \cdot x_2 \cdot x_1 \cdot x_0$
apellido1 $\equiv y_9 \cdot y_8 \cdot y_7 \cdot y_6 \cdot y_5 \cdot y_4 \cdot y_3 \cdot y_2 \cdot y_1 \cdot y_0$
apellido2 $\equiv z_9 \cdot z_8 \cdot z_7 \cdot z_6 \cdot z_5 \cdot z_4 \cdot z_3 \cdot z_2 \cdot z_1 \cdot z_0$



Fulano Zutánez Doe,
nombre ≡ 0 **10101011**
apellido1 ≡ 1 **0111110100**
apellido2 ≡ 000101 **1010**

101011 = 43
110100 = 52
1010 = 10
10 = 2
011 = 3

Posición:
$52 + 2^6 \cdot 10 + 2^{10} \cdot 43 + 2^{16} \cdot 3 + 2^{19} \cdot 2 =$ **1158836**

© 2008 LaBDa – Universidad Carlos III Madrid

Tema 6.5.3: Organizaciones Multiclave

Algunos Ejemplos:

- El directorio d-dimensional forma un cuadriculado del universo (***rejilla***)
- Es necesario almacenar la forma de la rejilla (*escala de la rejilla)*
- *El fichero así organizado es un* **Fichero Rejilla** (Grid File)

- También es posible hacer una rejilla regular: **EXCELL** (Extendible CELL)
- Esto evitaría su almacenamiento (y mantenerla en memoria principal)
- Pero si desborda y se escinde un cluster, el tamaño del directorio aumenta 2^l veces. El área de datos sólo aumenta lo necesario porque sólo se escinde el cluster necesario, dejando el resto de entradas del directorio apuntando a los mismos clusters.

- Si tomamos en una rejilla formada por intervalos no rectangulares y anidados almacenada en un árbol balanceado, tenemos un **Fichero BANG** (Balanced and Nested Grid)

© 2008 LaBDa – Universidad Carlos III Madrid