



UA

Unidad 2: Procesamiento y Optimización de Consultas

*Bases de Datos Avanzadas, Sesión 8:
Algoritmos de Procesamiento de Consultas y Coste*

*Iván González Diego
Dept. Ciencias de la Computación
Universidad de Alcalá*



UA

INDICE

- *Introducción.*
- *Operación de Proyección*
- *Operación de selección*
- *Ordenación*
- *Operación de reunión*
- *Otras operaciones*
- *Materialización*
- *Encauzamiento*

Referencias: *Silberschatz 4^a Ed.*
Elmasri, 3^a Ed.



UA

Operación de Proyección

- *Sólo búsqueda lineal. Explora cada bloque del fichero y elimina los campos que no se utilizan*
 - *Coste estimado $b_r \Rightarrow$ número de bloques de una tabla*



UA

Operación de Selección

- *Algoritmo A1 \Rightarrow búsqueda lineal. Explora cada bloque del fichero y verifica todos los registros para ver si se satisface la condición.*
 - Coste estimado $b_r \Rightarrow$ número de bloques explorados
 - Selección sobre un atributo clave, coste medio $\Rightarrow b_r / 2$
 - Para al encontrar el registro.
 - Búsqueda lineal, se puede aplicar a:
 - Condiciones de selección.
 - Ordenación de registros
 - Disponibilidad de índices.



UA

Operación de Selección

- *Algoritmo A2 ⇒ búsqueda binaria.*
 - Archivo ordenado según un atributo
 - Condición de selección ⇒ comparación de igualdad
 - Se asume que los bloques de una relación se almacenan contiguamente
 - Coste estimado (*nº de bloques de disco a explorar*):
 - $\lceil \log_2 b_r \rceil$ ⇒ coste de localizar la primera tupla por una búsqueda binaria de los bloques.
 - $+ \lceil n_{rc} / f_R \rceil$ ⇒ *nº de bloques que contienen registros que satisfacen la condición.*
 - -1 (*de localizar el primer bloque*)

$$\text{Coste} = \lceil \log_2 b_r \rceil + \lceil n_{rc} / f_R \rceil - 1$$



UA

Selecciones usando índices

- *Exploraciones índices \Rightarrow algoritmos de búsqueda que utilizan índices*
 - *La condición de selección debe estar en el campo de búsqueda del índice*
- *Coste = Coste del índice + Coste de buscar en el archivo de datos*
- *Depende:*
 - *Tipo de índice: Secuencial, B+, Hash, Rejilla, Mapa de Bit....*
 - *El índice puede ser primario o secundario sobre un campo clave o no.*
 - *Archivo de datos: si está ordenado por el campo de búsqueda o no*
 - *Es posible que sea más costoso que la búsqueda secuencial.*



UA

Ordenación

- *Se puede construir un índice en la relación \Rightarrow utilizar el índice para leer \Rightarrow puede hacer leer un bloque por cada tupla*
- *Para relaciones que caben en memoria \Rightarrow técnicas de ordenación clásicas (QuickSort) \Rightarrow Coste = 0 si no se lee del disco la tabla*
- *Para relaciones que no caben en memoria \Rightarrow ordenación externa
 - Ordenación-mezcla externa es un algoritmo muy utilizado.*



UA

Ordenación – Mezcla externa

- $M \Rightarrow$ tamaño de la memoria intermedia
- 1. \Rightarrow Crear secuencias ordenadas. $i=0$
 - Repetir hasta el fin de la relación
 - (a) Leer M bloques de la relación en memoria
 - (b) Ordenar la parte de la relación en memoria
 - (c) Escribir los datos ordenados al archivo de secuencias R_i
 - El valor final de i es N
- 2. \Rightarrow Mezclar las secuencias (Tamaños de N). Asumir $N < M$
 - 1. Usar N páginas de memoria para leer las secuencias y 1 bloque para la salida. Leer el primer bloque de cada secuencia en su página de memoria
 - 2. Repetir
 - 1 Seleccionar el primer registro (según el orden) entre todas las páginas
 - 2 Escribir el registro al buffer de salida. Si se llena \Rightarrow escribir disco
 - 3 Borrar el registro de su buffer de entrada. Si el buffer se vacía \Rightarrow leer el siguiente bloque en el buffer de entrada
 - 3. Hasta todos los buffers de entrada están vacíos:



UA

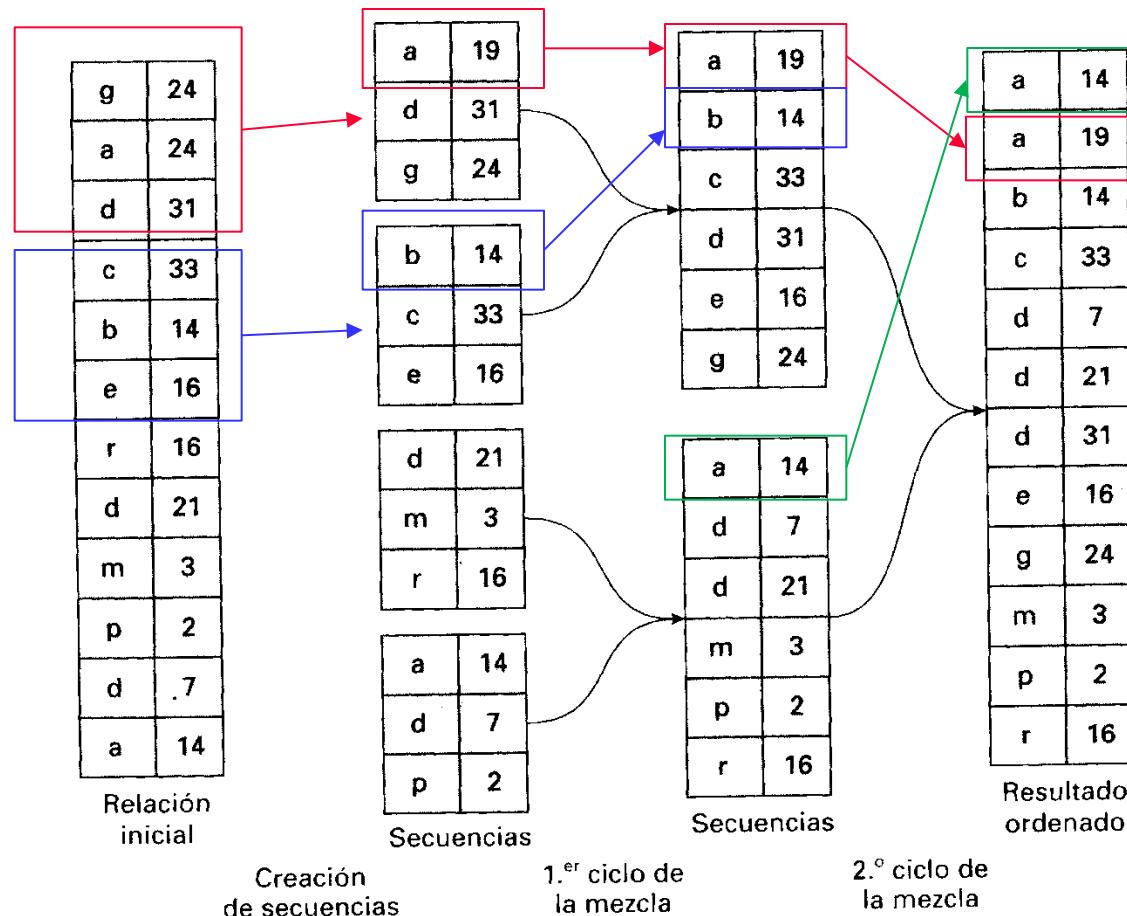
Ordenación – Mezcla externa

- *Si $N \geq M$, se requieren varios estados de mezcla*
 - *En cada paso, grupos contiguos de $M-1$ secuencias se mezclan*
 - *Un paso reduce el número de secuencias en un factor de $M-1$ y crea secuencias más grandes del mismo factor*
 - *Ejemplo: Si $M=11$ y hay 90 secuencias, 1 paso reduce el número de secuencias a 9 , 10 veces el tamaño de la secuencia inicial*
 - *Los pasos se repiten hasta que todas las secuencias se han mezclado en 1*



UA

Ordenación – Mezcla externa: Ejemplo



*M=3 bloques, cada bloque 1 registro,
en cada paso se lee y escribe salvo la última salida*



UA

Ordenación – Mezcla externa: Ejemplo

Análisis del Coste

- Total número de pasos requeridos: $\lceil \log_{M-1}(b_r/M) \rceil$
- Acceso al disco para la creación de la secuencia inicial, así como en cada paso es $2b_r$
 - Para el paso final no se cuenta el coste de escritura, ya que puede producir la ordenación como resultado sin escribir.
- El número total de accesos al disco es:

$$b_r (2 \lceil \log_{M-1}(b_r / M) \rceil + 1)$$

- Caso anterior: $M=3$, $b_r=12$, Coste = $12 * (2 \lceil \log_{3-1}(12 / 3) \rceil + 1) = 60$ bl.
- La salida de la operación no se graba, si se grabase en disco: $60+12=72$ bl.



UA

Operación Reunión

- *Diferentes algoritmos para implementar las reuniones:*
 - *Bucle anidado por bloques*
 - *Bucle anidado indexado*
 - *Reunión por mezcla*
 - *Reunión por asociación.*
- *La elección se basa en la estimación del coste*
- *Ejemplos:*
 - *Número de registros de cliente: 10.000 , impositor: 5000*
 - *Número de bloques de cliente: 400 , impositor 100*

impositor \bowtie cliente



UA

Bucle Anidado por bloques

- Variante del bucle anidado en la cuál se utilizan bloques en vez de tuplas

```
for each bloque Br de r do begin  
    for each bloque Bs de s do begin  
        for each tupla tr de Br do begin  
            for each tupla ts de Bs do begin  
                Verificar si (tr,ts) satisface la condición  
                Si lo cumple, añadir tr • ts al resultado.  
            end  
        end  
    end  
end
```



UA

Bucle Anidado por bloques

- Peor caso: $b_r * b_s + b_r$ bloques. $M=3$
 - Cada bloque de la relación interior se lee una vez para cada bloque de la relación externa (en vez de una vez para cada tupla)
- Mejor caso: $b_r + b_s$ bloques. $M \geq \min\{b_r, b_s\} + 2$
- Caso General:
 - En bucle anidado por bloques, usar $M - 2$ bloques de disco como unidad para la relación externa,
 $M \Rightarrow$ tamaño de la memoria en bloques. Los otros 2 para la relación interna y la salida

$$\text{Coste} = \lceil b_r / (M-2) \rceil * b_s + b_r$$



UA

Bucle Anidado por bloques

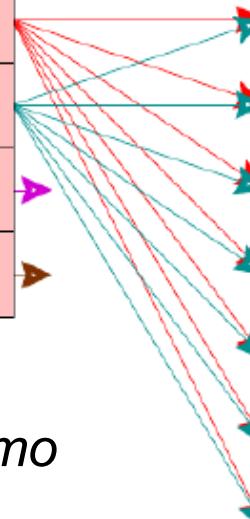
Table 1

aag
aay
aar
aai

Table 2

aai
aag
aas
aar
aay
aaa
aag

Algoritmo



Entrada r

aag
aay
aar

Salida



Memoria M=5

Entrada s

aai

aag
aay
aar
aai

Disco

aai
aag
aas
aar
aay
aaa
aag

$$\text{Coste} = \lceil b_s / (M-2) \rceil * b_s + b_r = 2 * 7 + 4 = 16 \text{ bl.}$$

Este caso factor bloque = 1 reg/bl



UA

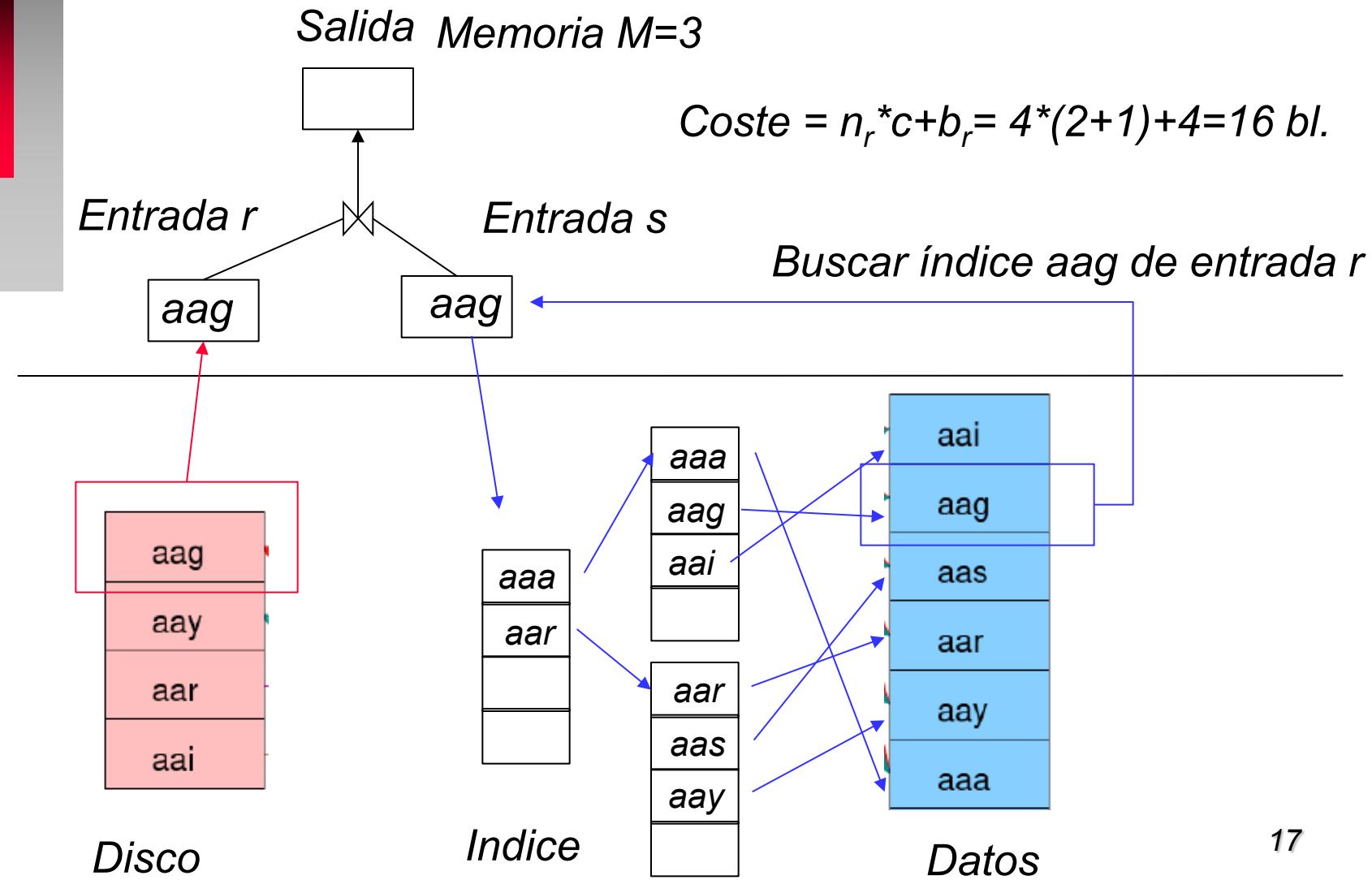
Bucle Anidado Indexado

- *Búsquedas con índices sustituyen a la exploración en ficheros*
 - *Reunión es una equi-reunión o reunión natural*
 - *Se dispone de un índice en un atributo de la relación interna*
 - *Se puede construir un índice temporal*
- *Para cada tupla t_r de la relación externa r, usar el índice para buscar tuplas en s que satisfacen la condición con la tupla t_r .*
- *Peor caso: buffer tiene sólo espacio para una página de r, y, para cada tupla de r, se realiza una búsqueda indexada en s.*
- *Coste: $b_r + n_r * c$*
 - *Donde c es el coste de una única selección en s utilizando la condición*
- *Si hay índices disponibles para r y s, es más eficiente utilizar como relación externa la que tiene menos tuplas*



UA

Bucle Anidado Indexado





UA

Ejemplo de Costes de Bucles anidados

- *Analizar impositor \bowtie cliente, con impositor como relación externa*
- *Cliente tiene un índice primario B^+ en el atributo nombre-cliente , con 20 entradas en cada nodo índice*
- *Como cliente tiene 10,000 tuples, la altura del árbol es 4 y 1 acceso más se necesita para encontrar el dato real*
- *Impositor tiene 5000 tuplas*
- *Coste de bucle anidado por bloques*
 - $400 * 100 + 100 = 40,100$ accesos a disco asumiendo el peor caso
- *Coste de bucle anidado indexado*
 - $100 + 5000 * 5 = 25,100$ accesos al disco



UA

Reunión por mezcla

1. Ordenar ambas relaciones por sus atributos en común, si no lo están
2. Mezclar las relaciones ordenadas para unirlas
 1. Paso de unión \Rightarrow similar al paso de mezcla del algoritmo ordenación – mezcla
 2. Principal diferencia \Rightarrow manejo de valores duplicados

The diagram illustrates the merging of two relations, r and s , into a third relation t . Relation r is shown as a table with columns $a1$ and $a2$. Relation s is shown as a table with columns $a1$ and $a3$. An arrow labeled "pr" points from r to s , indicating a projection operation. Another arrow labeled "ps" points from s to the final merged relation t .

	$a1$	$a2$
a		3
b		1
d		8
d		13
f		7
m		5
q		6

	$a1$	$a3$
a		A
b		G
c		L
d		N
m		B



UA

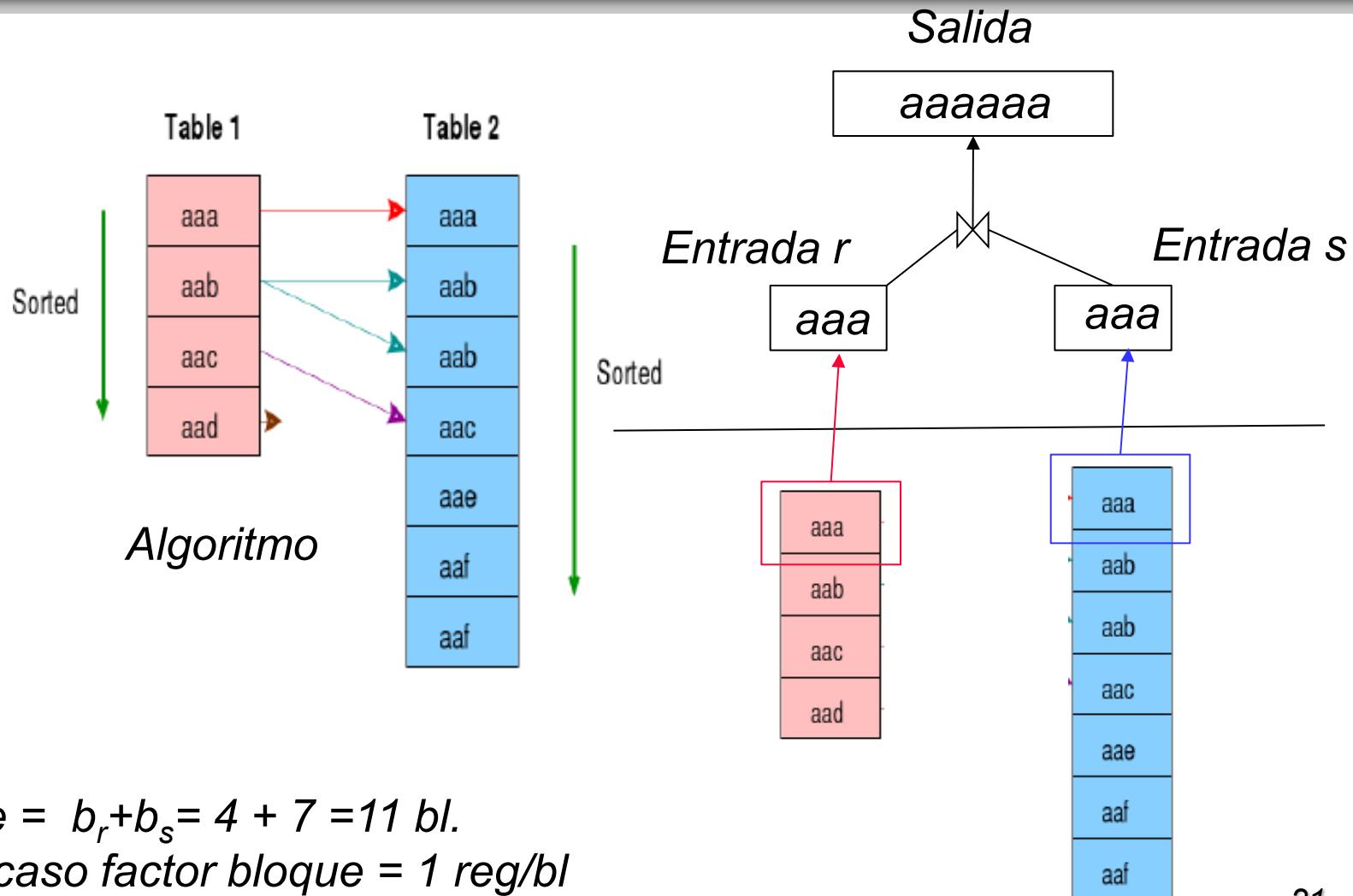
Reunión por mezcla

- *Se puede utilizar para equi-reuniones y reuniones naturales*
- *Cada bloque se necesita leerlo una sola vez (asumiendo que todas las tuplas para un valor dado de los atributos de la reunión se encuentran en memoria)*
- *Número de bloques de acceso:*
 $b_r + b_s + \text{coste de ordenar las relaciones (si se necesita).}$



Reunión por mezcla

UA





UA

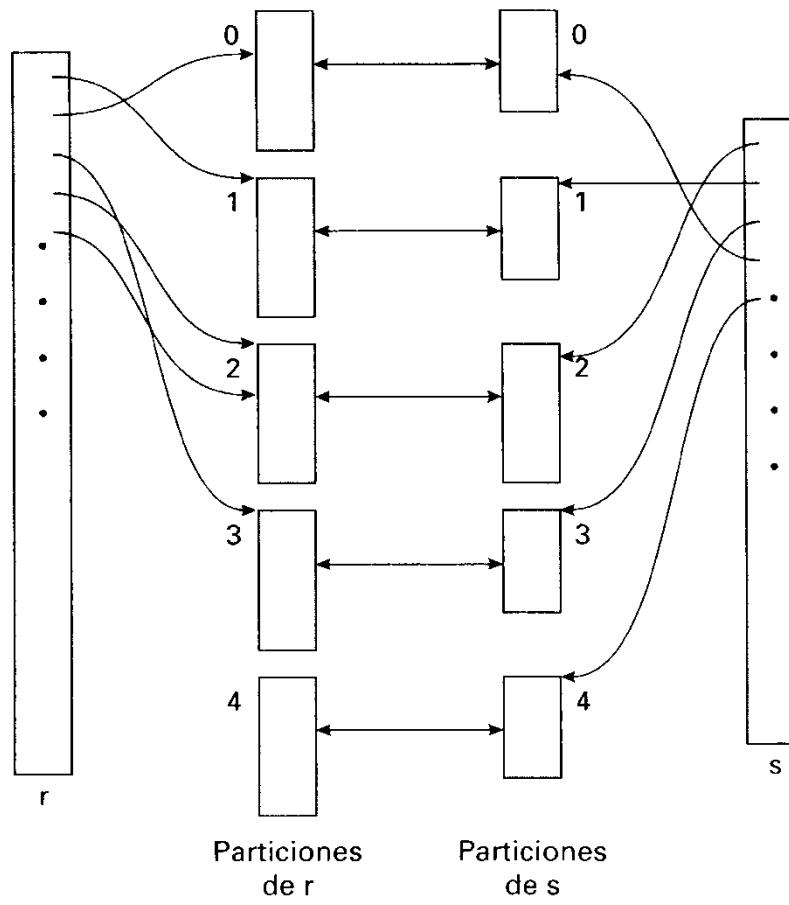
Reunión por asociación

- *Se aplica para equi-reuniones y reuniones naturales*
- *Una función de asociación h se usa para dividir las tuplas de ambas relaciones*
- *h asigna valores $\{0, 1, \dots, n\}$ a AtributosR, donde AtributosR son los atributos comunes de r y s usados en la reunión natural.*
 - r_0, r_1, \dots, r_n denota particiones de las tuplas de r
 - cada tupla $t_r \in r$ se coloca en la partición r_i donde $i = h(t_r [AtributosR])$.
 - s_0, s_1, \dots, s_n denota particiones de las tuplas de s
 - cada tupla $t_s \in s$ se coloca en la partición s_i , donde $i = h(t_s [AtributosR])$.



UA

Reunión por asociación





UA

Reunión por asociación

- r tuplas en r_i se necesitan comparar con s tuplas en s_i
- No se necesita comparar con s tuplas en otra partición debido a:
 - Una tupla r y una tupla s que satisfacen la condición de reunión, tendrán el mismo valor para los atributos de reunión.
 - Si ese valor es asociado a un valor i , la tupla r tiene que estar en r_i la tupla s_i .



UA

Reunión por asociación

- *La reunión por asociación se realiza:*
 1. *Partitionar la relación s, usando la función h. Un bloque de memoria se reserva como salida de cada partición.*
 2. *Partitionar la relación r*
 3. *Para cada i:*
 - (a) *Cargar s_i en memoria y construir un índice hash usando el atributo de la reunión. (Este índice usa una función diferente que h)*
 - (b) *Leer las tuplas de r_i del disco una por una. Para cada tupla t_r, localizar la tupla t_s en s_i usando el índice hash de memoria. El resultado será la concatenación de sus atributos*

*La relación s se llama entrada para construir
y r entrada para probar*



UA

Ejemplo de reunión por asociación

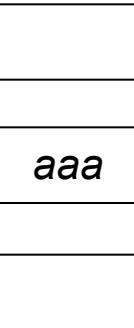
Disco

aaa
aab
aac
aad

Memoria

aaa

$h(x)$



Disco

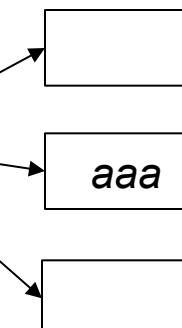
aab
aaa
aad

$h(x) N=3$ cajones

aaa
aab
aab
aac
aae
aaf
aaf

aaa

$h(x)$



Particiones 0-2 de r

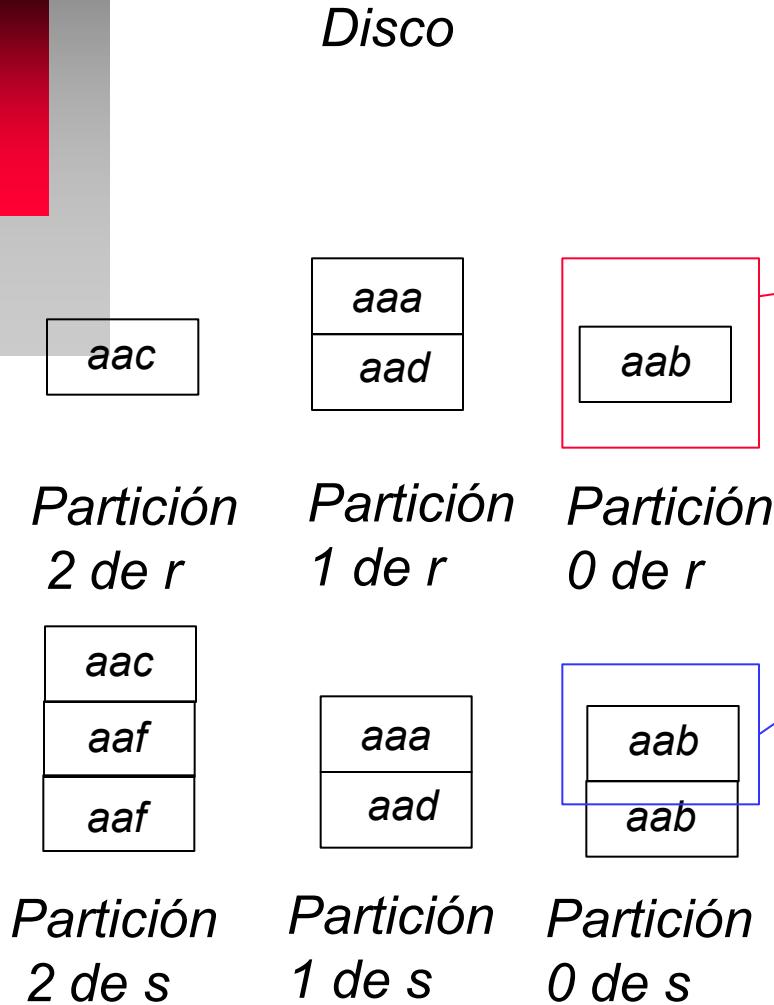
aab
aab
aaa
aad

Coste Construir: $br + br + bs + bs$



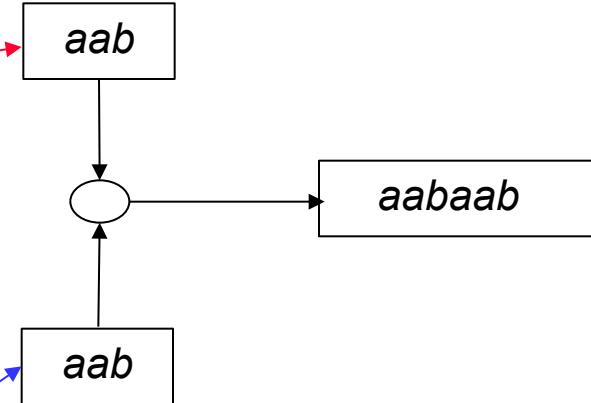
UA

Ejemplo de reunión por asociación



Memoria

Entrada cada partición de r completa



Entrada 1 bloque de s a mezclar

Coste mezclar: $br + bs$
Coste total: $3(br+bs)$



Coste de la reunión por asociación: Ejemplo

cliente \bowtie *impositor*

- Asumir que el tamaño en memoria es de 20 bloques
- $b_{\text{impositor}} = 100 \text{ and } b_{\text{cliente}} = 400.$
- *Impositor se usa como relación de construcción. Se partitiona en 5 particiones de tamaño 20 bloques. Esta partición se realiza en 1 paso*
- *Cliente se divide en 5 particiones, de tamaño 80 y se realiza en un paso.*
- *Coste total: $3(b_r + b_s) = 3(100 + 400) = 1500$ bloques de transferencia*
 - *Se ignora el coste de escribir parcialmente los bloques llenos*



UA

Reunión por asociación híbrida

- *Útil cuando el tamaño de la memoria es grande y la entrada de construcción es más grande que la memoria*
- **Principal característica:**
 - **Mantener la primera partición de la relación de construcción en memoria**
 - *Ejemplo.: Con memoria de 25 bloques, impositor puede ser particionado en 5 particiones de tamaño 20 bloques*
 - *División de la memoria:*
 - *La primera partición ocupa 20 bloques de memoria*
 - *1 bloque se usa para la entrada y 4 bloques más para guardar las otras 4 particiones*
 - *Cliente se divide de manera similar en 5 particiones de tamaño 80 ⇒ usando la primera para probar*
 - *Coste ⇒ $3(80 + 320) + 20 + 80 = 1300$ bloques de transferencia en vez de 1500*
 - *Es más útil ⇒ $M >> \sqrt{b_s}$*



UA

Otras Operaciones

- **Eliminación de duplicados:** Se puede implementar por asociación ó ordenación
 - Los duplicados aparecerán a continuación unos de otros.
Optimización: duplicados se pueden eliminar durante la generación de secuencias así como en la etapa de reunión/mezcla
 - Asociación es similar ⇒ duplicados estarán en el mismo cajón



UA

Otras Operaciones: Agregación

- Se puede implementar de manera similar a la eliminación de duplicados → **Coste de ordenar**
 - Ordenación o asociación se puede utilizar para traer tuplas juntas en el mismo grupo ⇒ aplicar funciones agregadas a cada grupo
 - Optimización: combinar tuplas en el mismo grupo durante el proceso de generación y mezclas intermedias, analizando valores agregados parciales
 - Para count, min, max, sum ⇒ mantener valores agregados en las tuplas encontradas en el grupo
 - Para avg ⇒ mantener suma y cuenta, y dividir al final



Otras Operaciones: Operaciones sobre conjuntos

- Conjuntos de operaciones (\cup , \cap and $-$): se puede usar una variante de mezcla después de ordenar ó una variante de asociación.
- **Coste como un join. → Ojo con resta, no es conmutativa**
- Ejemplo: Operaciones sobre conjuntos usando asociación
 1. Particionar ambas relaciones usando la misma función de asociación, creando, r_0, r_1, \dots, r_n , y $s_0, s_1, s_2, \dots, s_n$
 2. Procesar cada partición I : usando una función hash diferente, construir un índice asociativo en memoria para r_i
 3. – $r \cup s$: Añadir tuplas al índice asociativo s_i si no estaban ya. Añadir las tuplas del índice asociativo al resultado.
 - $r \cap s$: para cada tupla de s_i , probar el índice asociativo y pasar la tupla al resultado si estaba ya.
 - $r - s$: para cada tupla de s_i , si está en el índice asociativo, borrarla del índice. Añadir las tuplas restantes del índice asociativo al resultado.



UA

Otras Operaciones: Reunión externa

- **Reunión externa se puede calcular por**
 - Una reunión seguida añadiendo nulos a las tuplas que no participan.
 - Modificando los algoritmos de reunión.
- **Modificación de la reunión por mezcla $r \bowtie s$**
 - En $r \bowtie s$, las tuplas que no participan están en $r - \Pi_R(r \bowtie s)$
 - Modificar mezcla $r \bowtie s$: para cada tupla t_r de r que no cumple con la tupla de s , la salida de t_r se añade con nulos.
 - La reunión externa por la derecha y la reunión externa se hace similarmente.
- **Modificar la reunión por asociación $r \bowtie s$**
 - Si r es la relación prueba, las tuplas de r no coincidentes salen con valores nulos
 - Si r la relación de construcción, cuando se prueba la coincidencia de las tuplas de r con las de s , al final de s , la salida de las tuplas no coincidentes salen con nulos.
- **Coste como un Join**



UA

Ejemplo coste

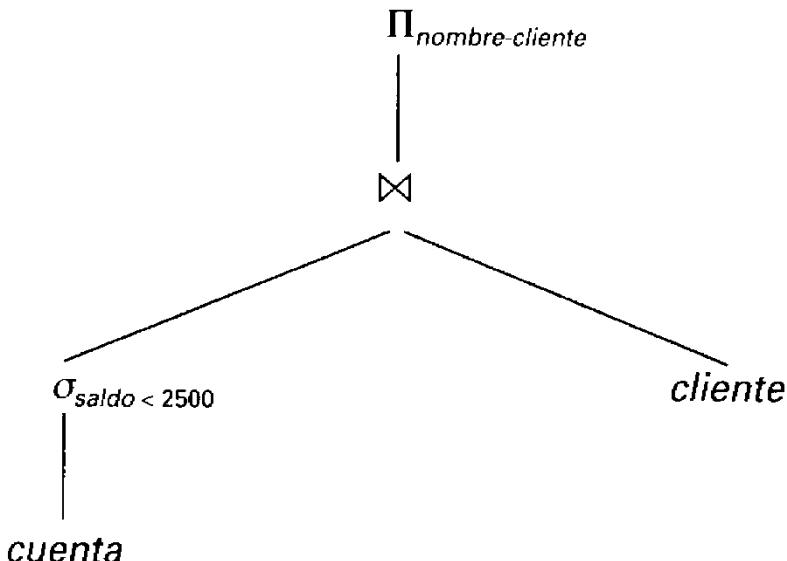
- Suponer esta consulta: $(r1 \bowtie r2) \bowtie r3$ donde
 - $r1$ tiene 10000 tuplas, 1000 bloques
 - $r2$ tiene 20000 tuplas, 2000 bloques
 - $r3$ tiene 30000 tuplas, 3000 bloques
 - $r1 \bowtie r2$ son 500 tuplas, 100 bloques
- Y suponer que las reuniones se realizan por medio de hash-join y que hay suficiente memoria para realizar la reunión. ¿Coste asociado?



Materialización

UA

- *Evaluación materializada: evalúa una operación a la vez, comenzando por el nivel más bajo.*
- *Usa resultados intermedios materializados en relaciones temporales para evaluar las operaciones del siguiente nivel*
- **Coste extra de grabar salida de operación**
- *Ejemplo:*





UA

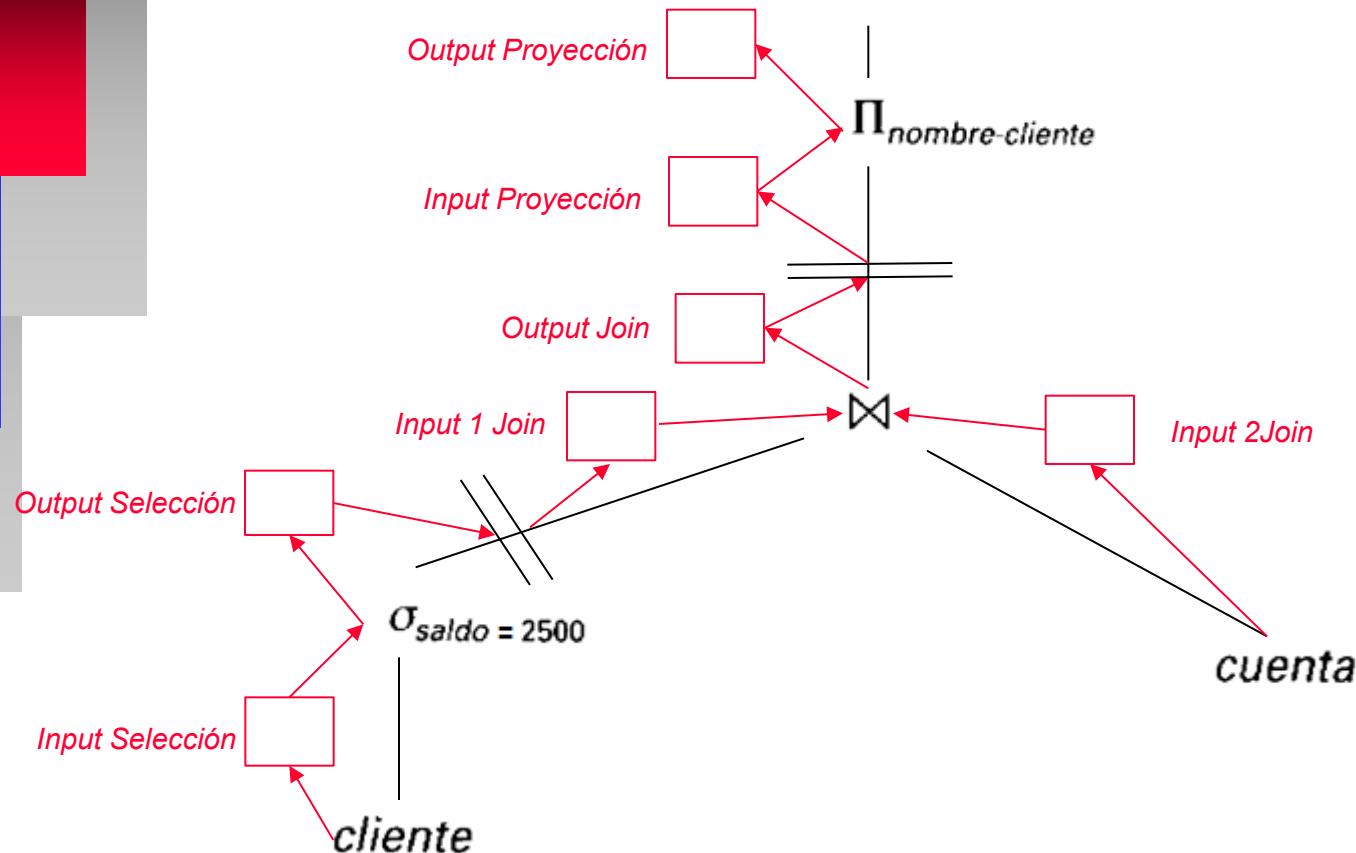
Materialización

- *Evaluación materializada siempre se puede aplicar*
- *Coste de escribir resultados al disco y leerlos puede ser muy alto*
 - *Las fórmulas de coste para las operaciones ignoran el coste de escribir resultados al disco, por lo que*
 - *Coste total = Suma del coste de operaciones individuales + coste de escribir resultados intermedios al disco*



UA

Materialización: Ejemplo



→ Bloque de memoria

→ Materializar Disco

Coste total con bucle anidado por bloques?



UA

Encauzamiento

- *Evaluación encauzada: evalúa varias operaciones a la vez, pasando los resultados de una operación a la siguiente.*
- *Ejemplo: en la expresión anterior, no almacenar*

$$\sigma_{saldo < 2500}(cuenta)$$

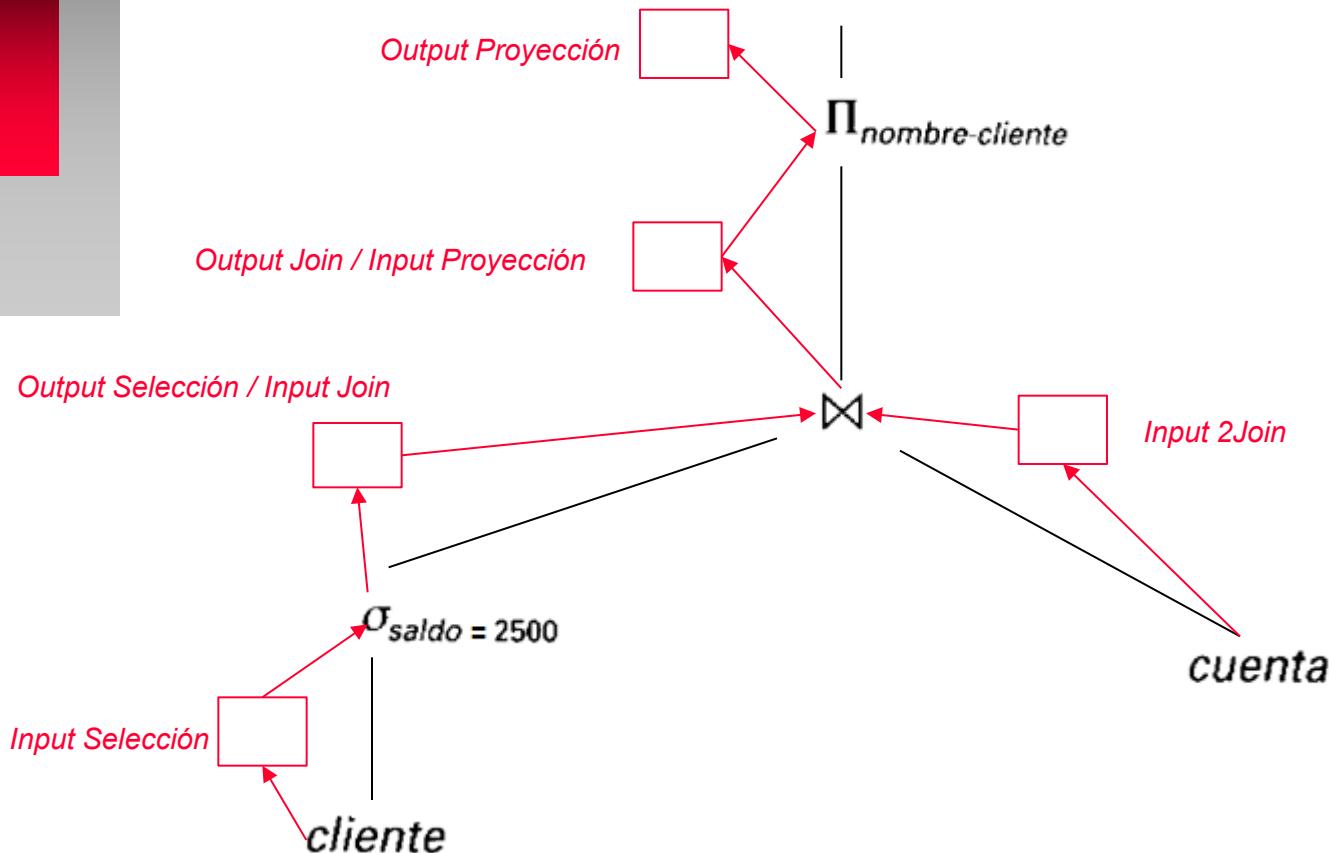
Pasa directamente las tuplas

- *Menos costosa que la materialización*
- *No siempre es posible: ordenación, reunión por asociación.*
- *Necesidad de algoritmos de evaluación que generen tuplas de salida incluso cuando se reciben tuplas de entrada.*
- *Se puede ejecutar de dos maneras:*
 - *Bajo demanda*
 - *Por los productores*



UA

Encauzamiento: Ejemplo



Coste total con bucle anidado por bloques?

→ Bloque de memoria

→ Materializar Disco



UA

Encauzamiento

Bajo demanda

- Sistema pide la siguiente tupla desde el nivel alto de operación.
- Cada operación pide la siguiente tupla a sus operaciones hijas para producir su siguiente tupla
- Entre llamadas \Rightarrow la operación debe de mantener el estado
- Se puede implementar como un iterador:
 - Open()
 - Exploración del fichero: inicializar, almacenar puntero
 - Reunión por mezcla: ordenar relaciones y almacenar punteros al comienzo de las relaciones ordenadas.
 - Next()
 - Salida de la siguiente tupla y actualizar puntero
 - Continuar con la mezcla desde el estado anterior hasta encontrar la siguiente tupla de salida. Actualizar puntero
 - Close()



UA

Encauzamiento

■ *Por producción:*

- *Los operadores producen tuplas impacientemente y las pasan a sus padres.*
 - *Se necesita memoria intermedia, para colocar las tuplas y los padres cogerlas.*
 - *Si la memoria se llena, el hijo espera hasta que haya espacio libre.*
- *Sistema cambia entre operaciones cuando tiene espacio en la memoria intermedia para generar tuplas.*