Unidad 2: Procesamiento y Optimización de Consultas

Bases de Datos Avanzadas, Sesión 6:
Algoritmos de Procesamiento de Consultas

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

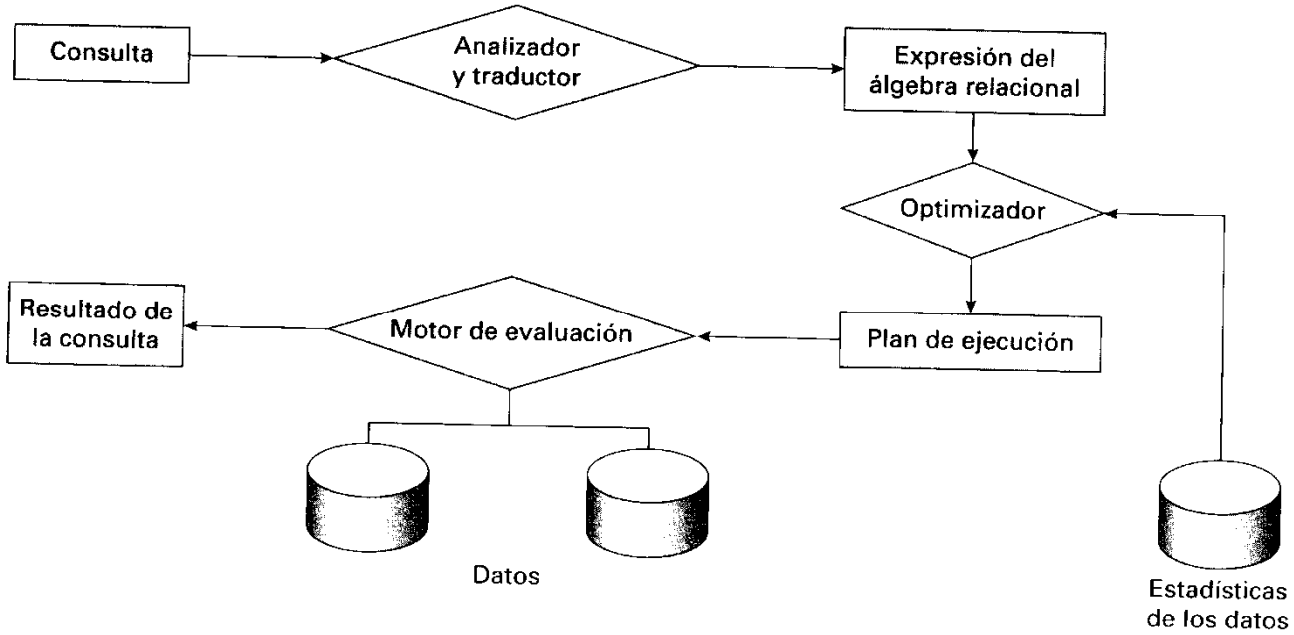
INDICE

- *Introducción.*
- *Medidas del Coste de una consulta*
- *Álgebra Relacional – SQL*
- *Operación de selección*
- *Ordenación*
- *Operación de reunión*
- *Otras operaciones*

Referencias: Silberschatz 4ª Ed. Pp 319 - 341
Elmasri, 3ª Ed. Pp 553 - 595

Pasos básicos en el procesamiento de consultas

- *1. Análisis y traducción.*
- *2. Optimización.*
- *3. Evaluación.*



Pasos básicos en el procesamiento de consultas

- *Análisis y traducción*
  - *Traducción de la consulta a su formato interno.*
  - *Se transforma en el álgebra relacional extendida.*
  - *Se verifica la sintaxis y se verifican las relaciones.*
- *Evaluación*
  - *El motor de ejecución de la consulta toma un plan de evaluación, ejecuta el plan y devuelve el resultado de la consulta.*

Pasos básicos en el procesamiento de consultas: Optimización

- *Un expresión del álgebra relacional puede tener muchas expresiones equivalente:*
  - *Ejemplo:* $\sigma_{\text{saldo} < 2500}(\Pi_{\text{saldo}}(\text{cuenta}))$ es equivalente a
    $\Pi_{\text{saldo}}(\sigma_{\text{saldo} < 2500}(\text{cuenta}))$
- *Cada operación del álgebra relacional puede ser evaluada usando uno de los diferentes algoritmos* $\Rightarrow$ *Primitivas*
  - *Una expresión del álgebra relacional puede ser evaluada de muchas maneras*
- *Una secuencia de operaciones que se pueden utilizar para evaluar una consulta* $\Rightarrow$ *plan de evaluación*
  - *Ejemplo: poder usar un índice en saldo para encontrar saldo < 2500*
  - *O realizar una búsqueda completa y descartar los saldos* ≥ 2500

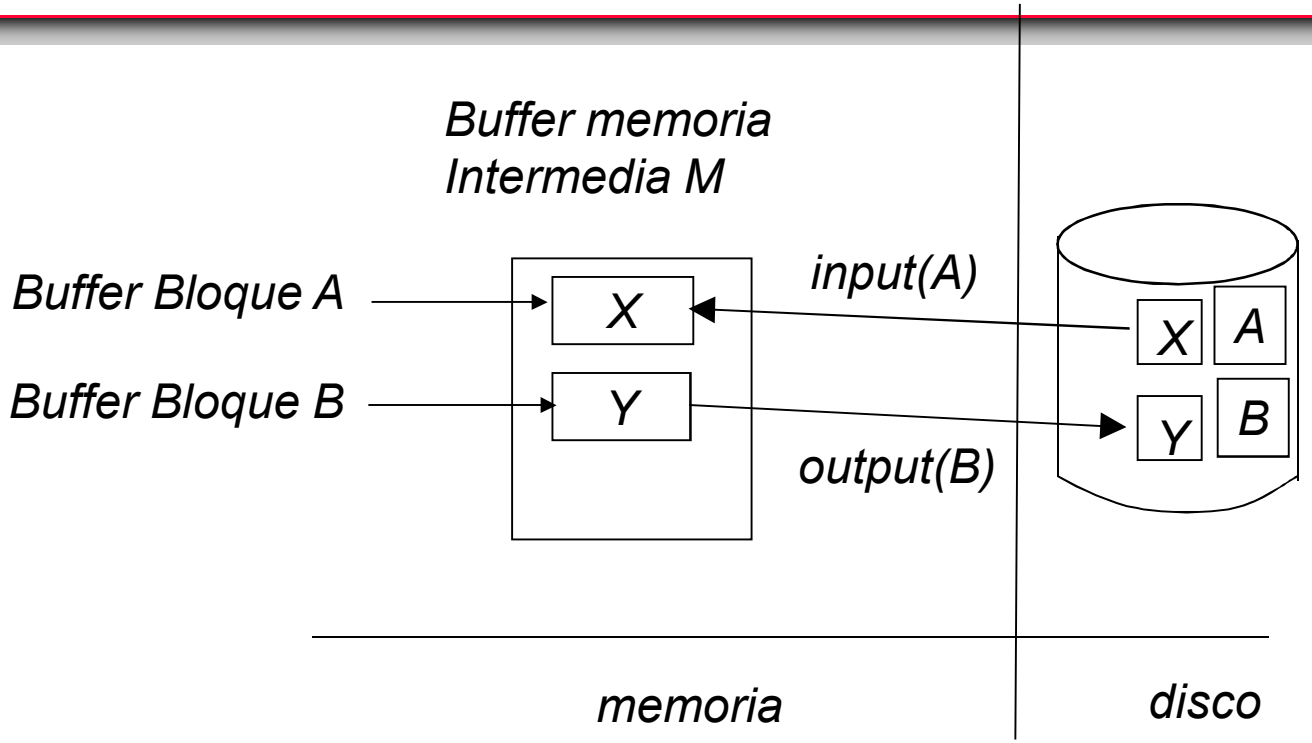
Pasos básicos en el procesamiento de consultas: Optimización

- *Optimización de consultas ⇒ entre todos los planes de evaluación de consultas, elegir la de menor coste*
  - *El coste se estima usando información estadística del catálogo ⇒ nº de tuplas de cada relación, tamaño de las tuplas, etc*
- *Medida de los costes de las consultas.*
- *Algoritmos para evaluar operaciones del álgebra relacional*
- *Combinación de algoritmos para evaluar expresiones*
- *Optimizar consultas ⇒ encontrar un plan de evaluación con el menor coste estimado.*

Medidas del coste de una consulta

- *Se mide por el tiempo utilizado para responder a la consulta*
  - *Factores: acceso al disco, CPU o red de comunicación.*
- *Coste más predominante: acceso disco*
  - *Fácil de estimar, teniendo en cuenta*
    - *Número de búsquedas ⇒ coste medio de búsqueda*
    - *Número de bloques leídos ⇒ coste medio de lectura de bloque*
    - *Número de bloques escritos ⇒ coste medio de escritura de bloque*
- *Se usará el número de bloques transferidos del disco como una medida del coste.*
- *Tamaño de la memoria disponible M (bloques/páginas en memoria)*
- *Lo que no cabe en RAM → se graba temporalmente en disco*

Medidas del coste de una consulta



Algebra Relacional - SQL

- Proyección $\pi_A(r)$ → SELECT DISTINCT A from r;
- Selección $\sigma_{condición}(r)$ → select * from r WHERE condición;
- Producto cartesiano r x s → select * from r,s;
- Unión r U s → select * from r UNION select * from s;
- Diferencia r – s → select * from r EXCEPT select * from s;
- Intersección r $\cap$ s → select * from r INTERSECT select * from s;
- Natural Join r $\bowtie$ s → select * from r NATURAL JOIN s;
- Join r $\bowtie_{condición}$ s → select * from r INNER JOIN s ON condición;
- Outer join r $\overset{\square}{\bowtie}$ s → select * from r FULL OUTER JOIN s;
- Agregado ${}_A g_{count(B)}(r)$ → select A,count(B) from r GROUP BY A;
- Select * from r ORDER BY A → no existe en Algebra Relacional

Operación de Selección

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

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)*

Selecciones usando índices

- *Exploraciones índices ⇒ algoritmos de búsqueda que utilizan índices*
  - *La condición de selección debe estar en la clave de búsqueda del índice*
- *A3 ⇒ Índice primario, condición igualdad en campo clave.*
  - *Obtiene un único registro que satisface la condición de igualdad*
  - *Coste ⇒ C_i+1*
- *A4 ⇒ Índice primario, igualdad basada en un campo no clave*
  - *Obtiene múltiples registros*
  - *Registros estarán en bloques consecutivos.*
  - *Coste ⇒ $C_i + \lceil n_{rc} / f_R \rceil$*
- *A5 ⇒ Índice secundario: igualdad*
  - *Obtiene un único registro si el campo de búsqueda es clave candidata*
    - *Coste ⇒ C_i+1*
  - *Obtiene varios registros si el campo de búsqueda no es campo clave*
    - *Coste ⇒ $C_i + n_{rc}$*
    - *Puede ser muy costosa*
    - *Cada registro puede estar en un bloque diferente ⇒ peor ⇒ un bloque / registro*

  Ci ⇒ Coste de buscar en el índice

Selecciones con condiciones de comparación

- *Se puede utilizar*
  - *Búsqueda lineal o búsqueda binaria*
  - *Usando índices*
- *A6 $\Rightarrow$ Indice primario, comparación. Relación ordenada en A*
  - *Para $\sigma_{A \geq V}$(r) usar índice para encontrar la primera tupla $\geq$ v explorar secuencialmente la relación desde allí.*
  - *Para $\sigma_{A \leq V}$ (r) explorar desde el principio del archivo hasta primera tupla > v. No se usa índice*
- *A7 $\Rightarrow$ Indice secundario, comparación*
  - *Para $\sigma_{A \geq V}$(r) usar el índice para encontrar la primera entrada en índice $\geq$ v y explorar el índice secuencialmente desde aquí, encontrando los punteros a los registros*
  - *Para $\sigma_{A \leq V}$ (r) buscar en los registros del índice hasta encontrar primera entrada > v*
  - *Hay que obtener los registros de datos de cada puntero.*
  - *Puede ser costosa.*

Implementación de selecciones complejas

- *Conjunción* $\Rightarrow \sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \theta n}(r)$
- *A8* $\Rightarrow$ *Selección conjuntiva utilizando un índice*
  - *Seleccionar una combinación de* θ_l *y algoritmos de A1 a A7 que proporcionen el menor coste para* $\sigma_{\theta i}$ (r)
  - *La operación se completa verificando las otras condiciones en memoria*
- *A9* $\Rightarrow$ *Selección conjuntiva utilizando un índice compuesto*
  - *Usar el índice multiclave si se puede utilizar en las condiciones*
- *A10* $\Rightarrow$ *Selección conjuntiva mediante la intersección de punteros*
  - *Requiere índices con punteros a registros.*
  - *Usar el correspondiente índice para cada condición y realizar la intersección de todos los conjuntos de punteros obtenidos.*
  - *Localizar los registros en el fichero*
  - *Si algunas condiciones no disponen de índices* $\Rightarrow$ *verificar en memoria*

Implementación de selecciones complejas

- *Disyunción* $\Rightarrow \sigma_{\theta 1 \vee \theta 2 \vee \ldots \theta n}(r)$

- *A11* $\Rightarrow$ *Selección disyuntiva mediante la unión de identificadores*
  - *Aplicable si todas las condiciones tienen índices disponibles*
    - Otro caso $\Rightarrow$ *usar exploración lineal*
  - *Usar el índice correspondiente para cada condición y tomar la unión de todos los conjuntos de registros de punteros*
  - *Localizar los registros del fichero con los punteros.*

- *Negación* $\sigma_{\neg\theta}(r)$
  - *Usar exploración lineal en el fichero*
  - *Si pocos registros satisfacen* $\neg\theta$ *y un índice se puede aplicar a* θ
    - *Encontrar los registros usando el índice y cargarlos del fichero.*

Ordenación

- *Se puede construir un índice en la relación ⇒ utilizar el índice para leer ⇒ puede hacer leer un bloque por cada tupla*
- *Para relaciones que caben en memoria ⇒ técnicas de ordenación clásicas (QuickSort)*
- *Para relaciones que no caben en memoria ⇒ ordenación externa*
  - *Ordenación-mezcla externa es un algoritmo muy utilizado.*

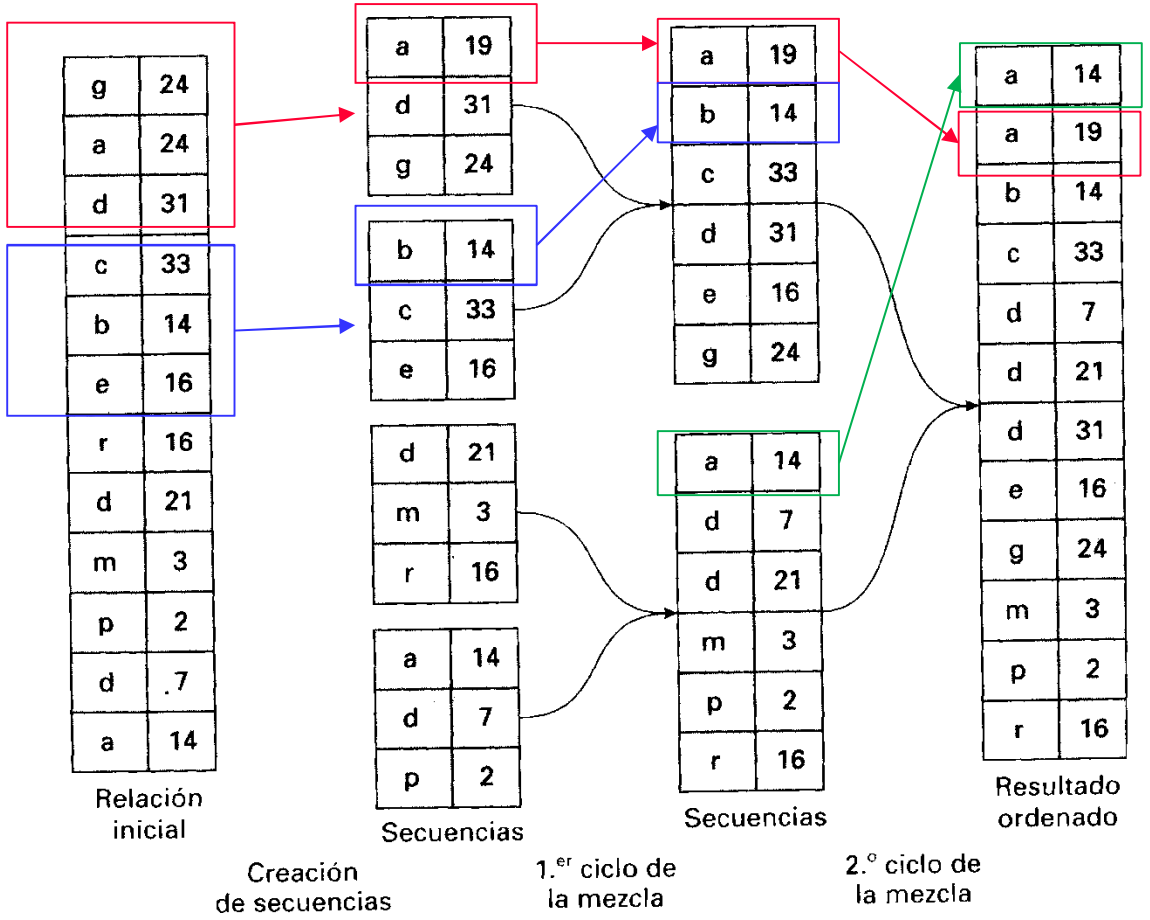
Ordenación – Mezcla externa

- *M ⇒ tamaño de la memoria intermedia*
- *1. ⇒ 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. ⇒ 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 ⇒ escribir disco*
    - *3 Borrar el registro de su buffer de entrada. Si el buffer se vacía ⇒ leer el siguiente bloque en el buffer de entrada*
  - *3. Hasta todos los buffers de entrada están vacíos:*

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*
    - *Ejplo: 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*

Ordenación – Mezcla externa: Ejemplo



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

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:

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

  - *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.*

Operación Reunión

- Diferentes algoritmos para implementar las reuniones:
  - Bucle anidado
  - 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

$$impositor \bowtie cliente$$

Bucle Anidado

- *Para realizar la reunión zeta:* r $\bowtie_\theta$ s

  for each tupla t_r en r do begin

    for each tupla t_s en s do begin

      comprobar que el par (tr,ts) satisface la condición de reunión θ

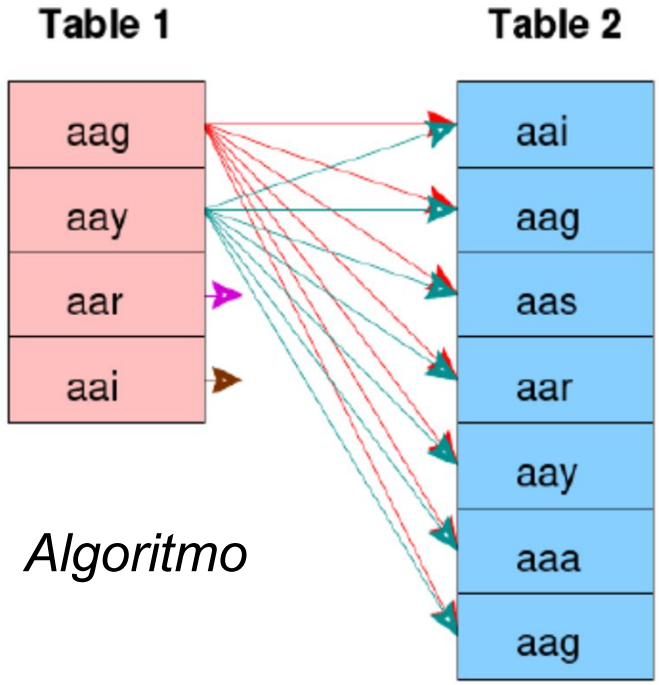
      Si se cumple $\Rightarrow$ *añadir* $t_r \bullet t_s$ al resultado

    end

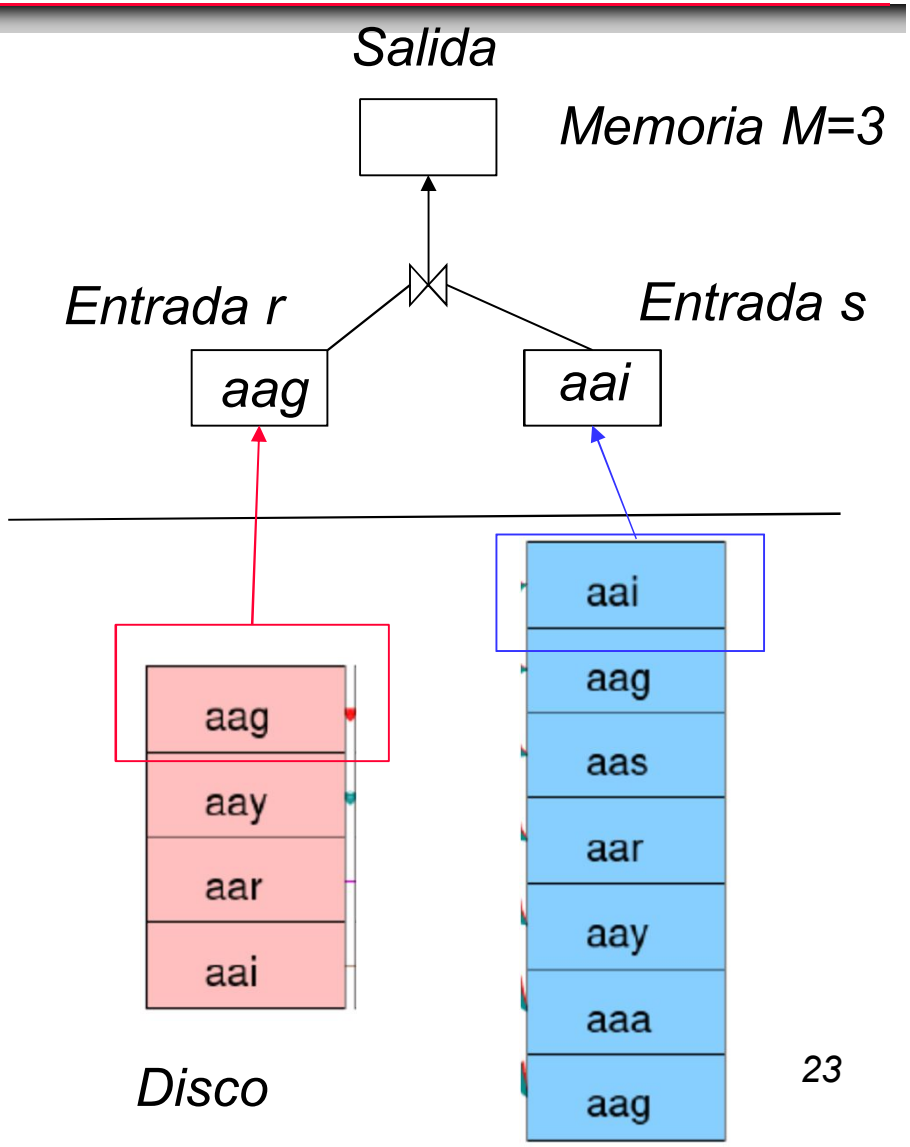
  end

- r se denomina relación externa y s relación interna
- No requiere índices y se puede utilizar con cualquier condición
- Costosa $\Rightarrow$ *examina cada par de tuplas de las dos relaciones*

Bucle Anidado por tuplas



Algoritmo



Disco

$Coste = n_r * b_s + b_r = 4*7+4=32$ bl.

Este caso factor bloque = 1 reg/bl

Bucle Anidado

- En el peor caso M=3, si hay suficiente memoria sólo para mantener un bloque de cada relación:

$$n_r * b_s + b_r \text{ accesos a disco}$$

- Si la relación cabe en memoria $\Rightarrow b_r + b_s$
- Asumiendo el peor coste:
  - 5000*400 + 100 = 2000100 accesos a disco con impositor como relación externa
  - 1000*100+400=1000400 accesos a disco con cliente como relación externa
- Si la relación impositor cabe en memoria, coste $\Rightarrow$ 500
- Es preferible usar reunión en bucle anidado por bloques

Bucle Anidado por bloques

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

for each bloque B_r *de* r *do begin*
  for each bloque B_s *de* s *do begin*
    for each tupla t_r *de* B_r *do begin*
      for each tupla t_s *de* B_s *do begin*
        Verificar si (t_r,t_s) *satisface la condición*
        Si lo cumple, añadir $t_r \bullet t_s$ *al resultado.*
      end
    end
  end
end

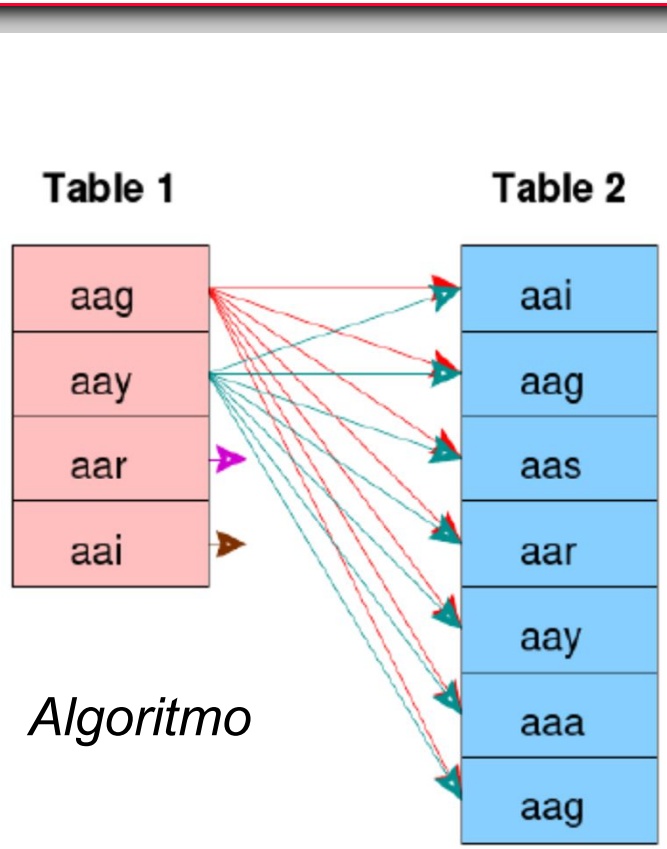
Bucle Anidado por bloques

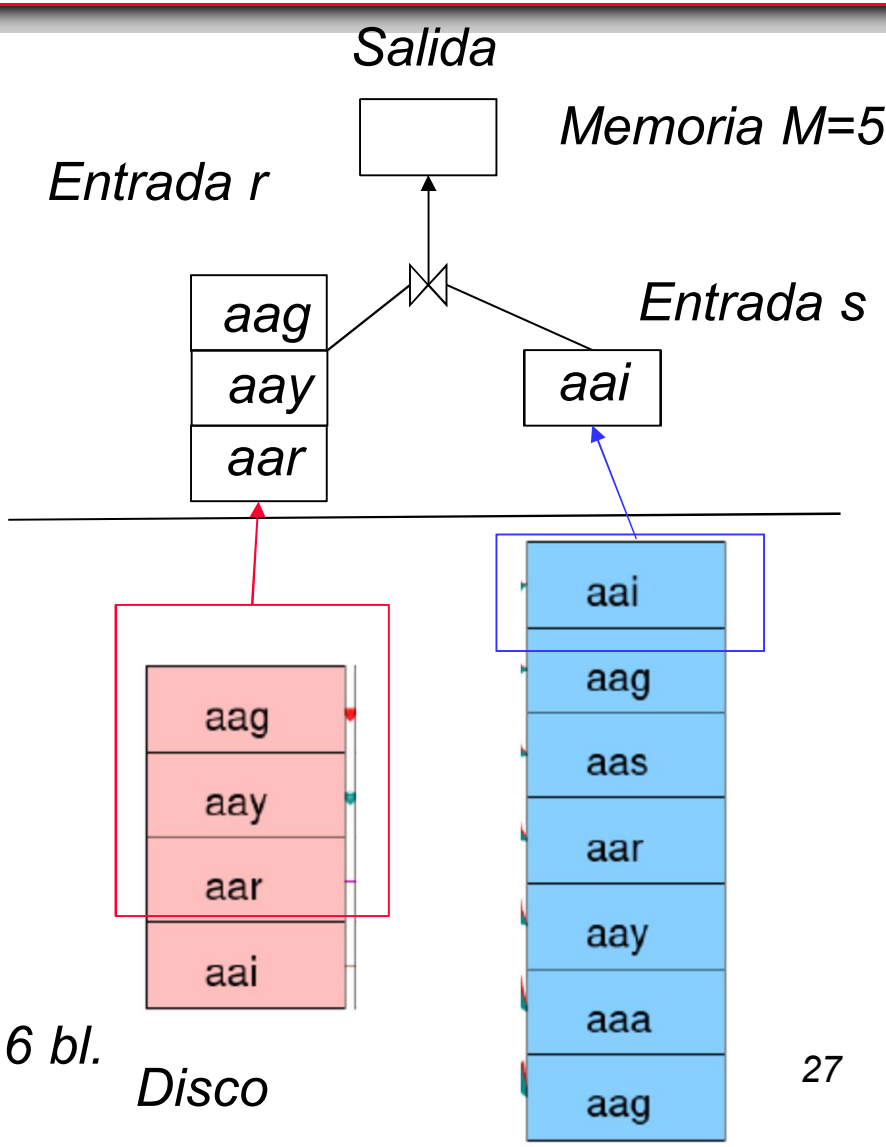
- *Peor caso:* $b_r * b_s + b_r$ *bloques.*
  - *Cada bloque de la relación interior s se lee una vez para cada bloque de la relación externa (en vez de una vez para cada tupla)*
- *Mejor caso:* $b_r + b_s$ *bloques.*
- *Mejoras de los bucles anidados:*
  - *En bucle anidado por bloques, usar M –2 bloques 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*

    $$Coste = \lceil b_r / (M-2) \rceil * b_s + b_r$$
  - *Si los atributos de la equi-reunión forman una clave de la relación interna, finalizar el bucle interno cuando se encuentra la primera coincidencia*
  - *Explorar el lazo interno alternativamente hacia adelante y atrás* $\Rightarrow$ *para usar los bloques restantes en el buffer (con LRU)*
  - *Usar índice de la relación interna si se dispone de él.*

Bucle Anidado por bloques



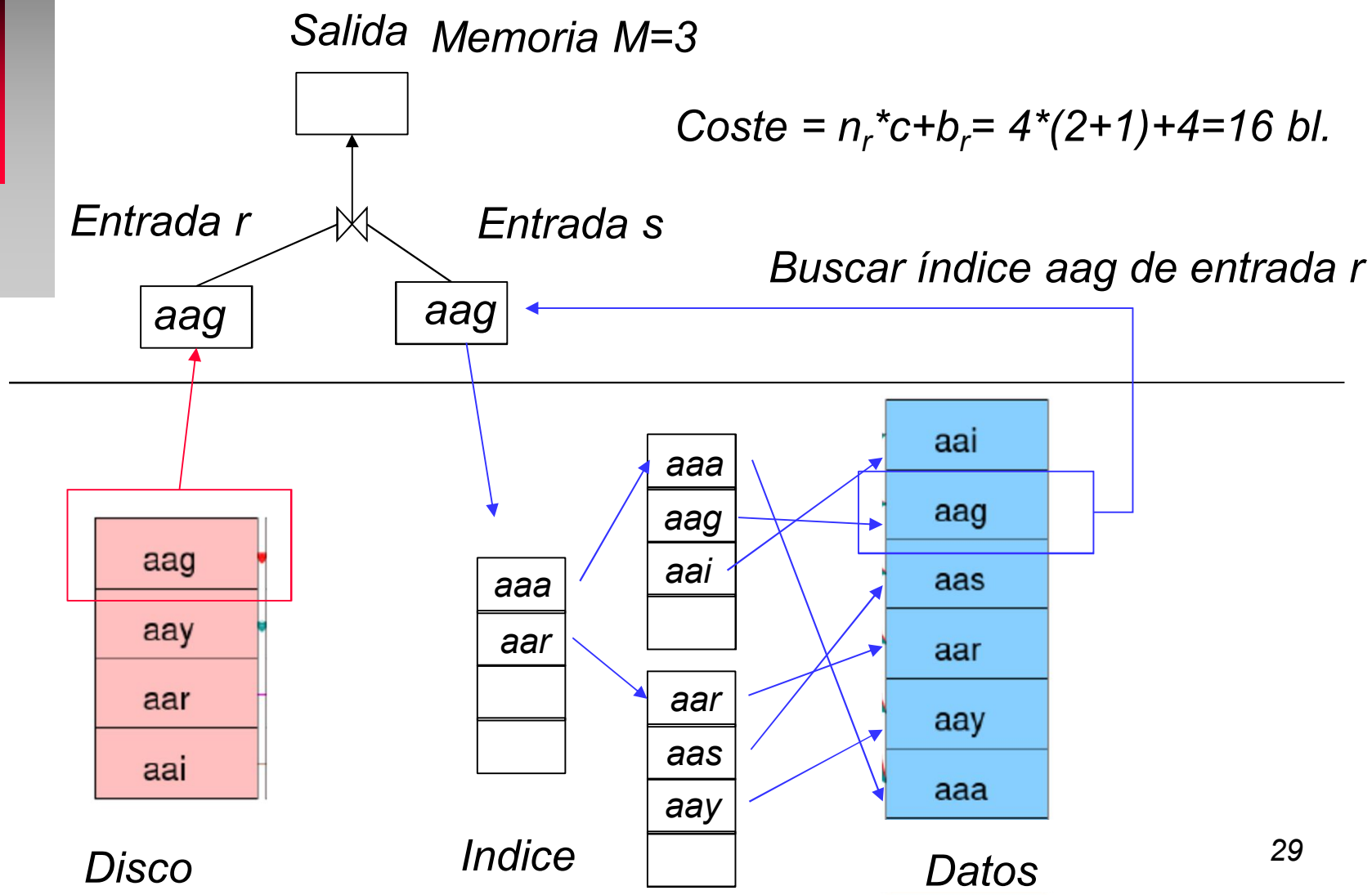


Coste = $\lceil b_r/(M-2) \rceil * b_s + b_r = 2*7+4=16$ bl.
Este caso factor bloque = 1 reg/bl

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*

Bucle Anidado Indexado

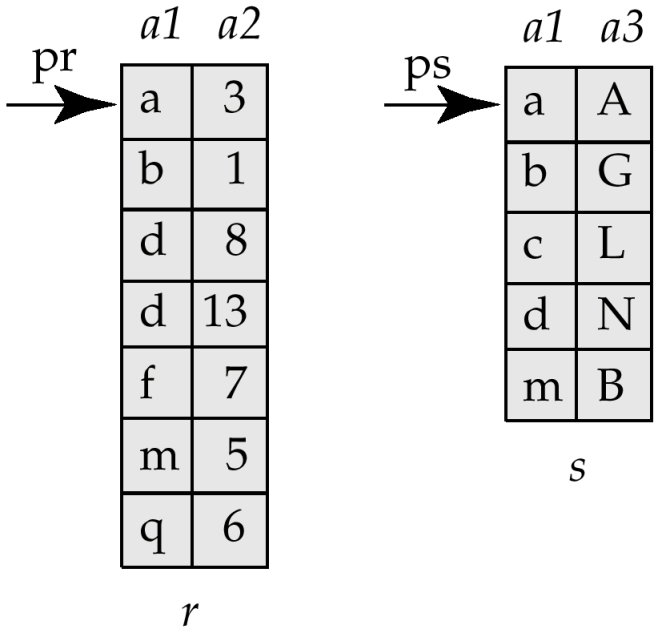


Ejemplo de Costes de Bucles anidados

- *Analizar* impositor ⋈ 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*

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 ⇒ similar al paso de mezcla del algoritmo ordenación – mezcla*
   2. *Principal diferencia ⇒ manejo de valores duplicados*

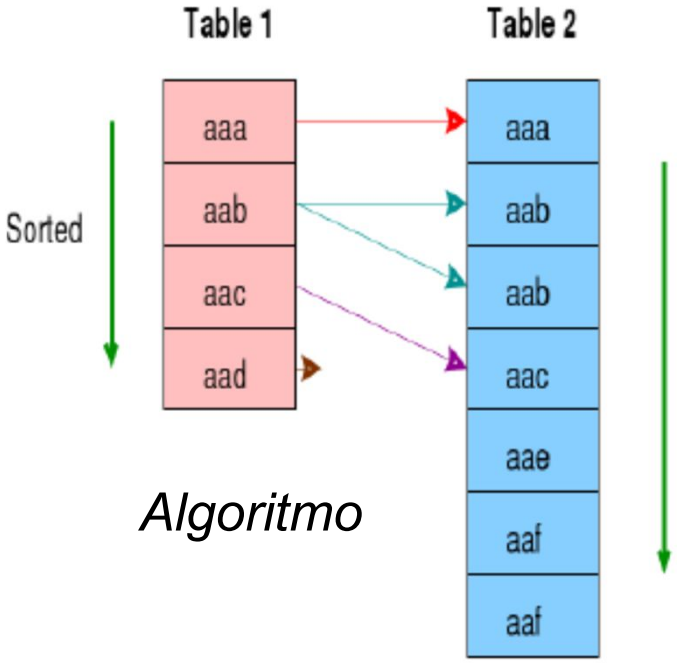


Reunión por mezcla

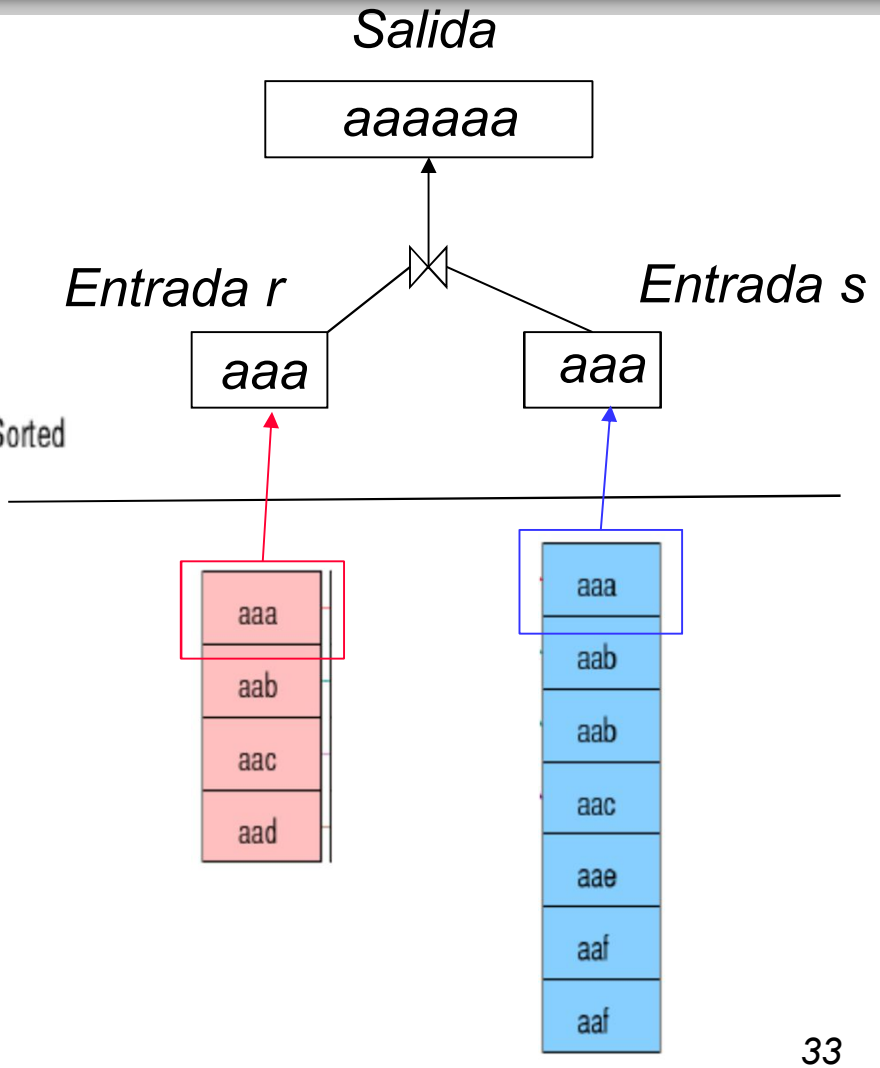
- *Se puede utilizar para equi-reuniones y reuniones naturales*
- *Cada bloque se necesita leerlo una sola vez (asumiendo que todas las tuplas para un valor dado de los atributos de la reunión se encuentran en memoria)*
- *Número de bloques de acceso:*

  $b_r + b_s$ + *coste de ordenar las relaciones (si se necesita).*
- *Reunión híbrida: Si una relación está ordenada y la otra tiene un índice secundario B^+ en el atributo de reunión:*
  - *Mezclar la relación ordenada con los nodos hojas del árbol*
  - *Ordenar el resultado según las direcciones de las tuplas de la relación desordenada*
  - *Explorar la relación desordenada, permitiendo una recuperación eficiente según el orden físico de almacenamiento, para completar la reunión.*

Reunión por mezcla



Algoritmo

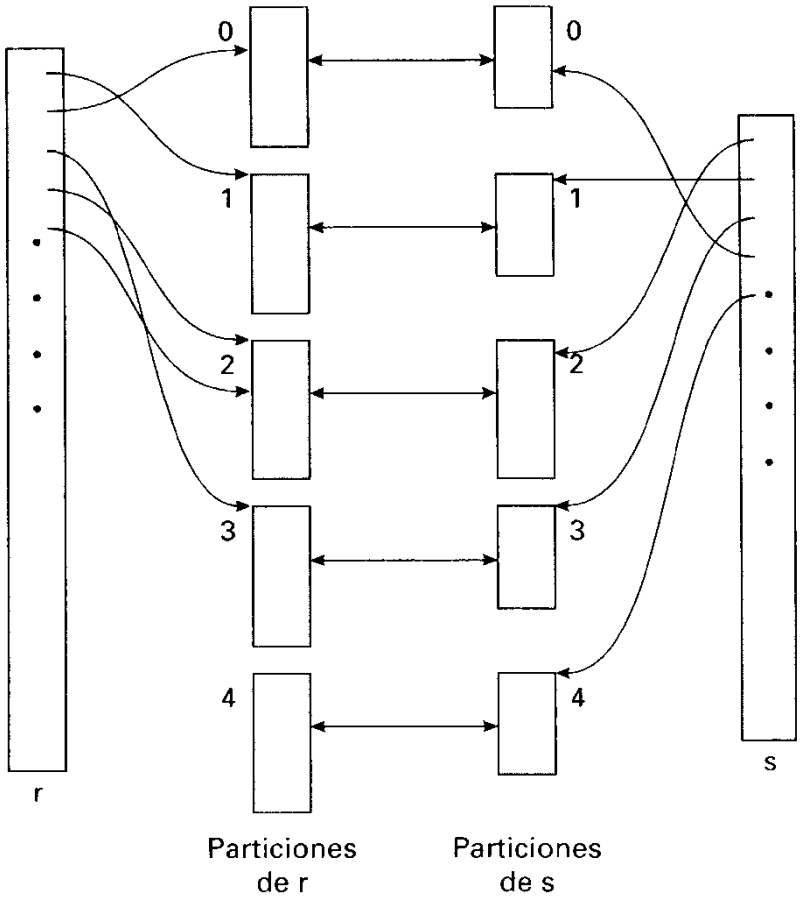


Coste = b_r+b_s= 4 + 7 =11 bl.
Este caso factor bloque = 1 reg/bl

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, ...,* n*} a AtributosR, donde* AtributosR *son los atributos comunes de* r *y* s *usados en la reunión natural.*
  - $r_0, r_1, \ldots, r_n$ *denota particiones de las tuplas de* r
    - *cada tupla* $t_r \in$ r *se coloca en la partición* r_i *donde* $i = h(t_r [AtributosR])$.
  - $s_0, s_1 \ldots, s_n$ *denota particiones de las tuplas de s*
    - *cada tupla* $t_s \in$s *se coloca en la partición* s_i*, donde* $i = h(t_s [AtributosR])$.

Reunión por asociación



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* s_i.

Reunión por asociación

- *La reunión por asociación se realiza:*

1. *Particionar la relación s, usando la función h. Un bloque de memoria se reserva como salida de cada partición.*

2. *Particionar la relación r*

3. *Para cada i:*

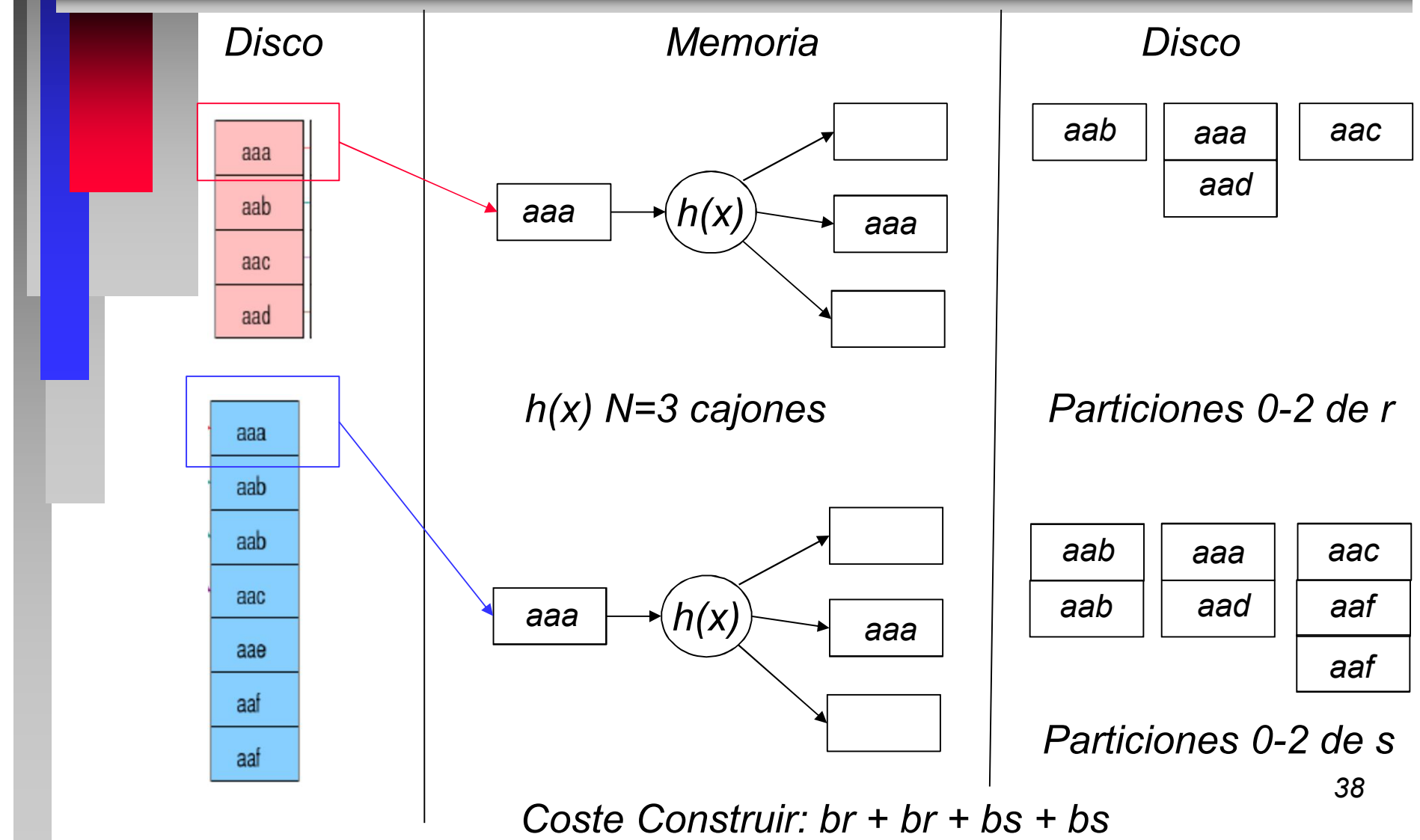
   (a) Cargar s_i en memoria y construir un índice hash usando el atributo de la reunión. (Este índice usa una función diferente que h)

   (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

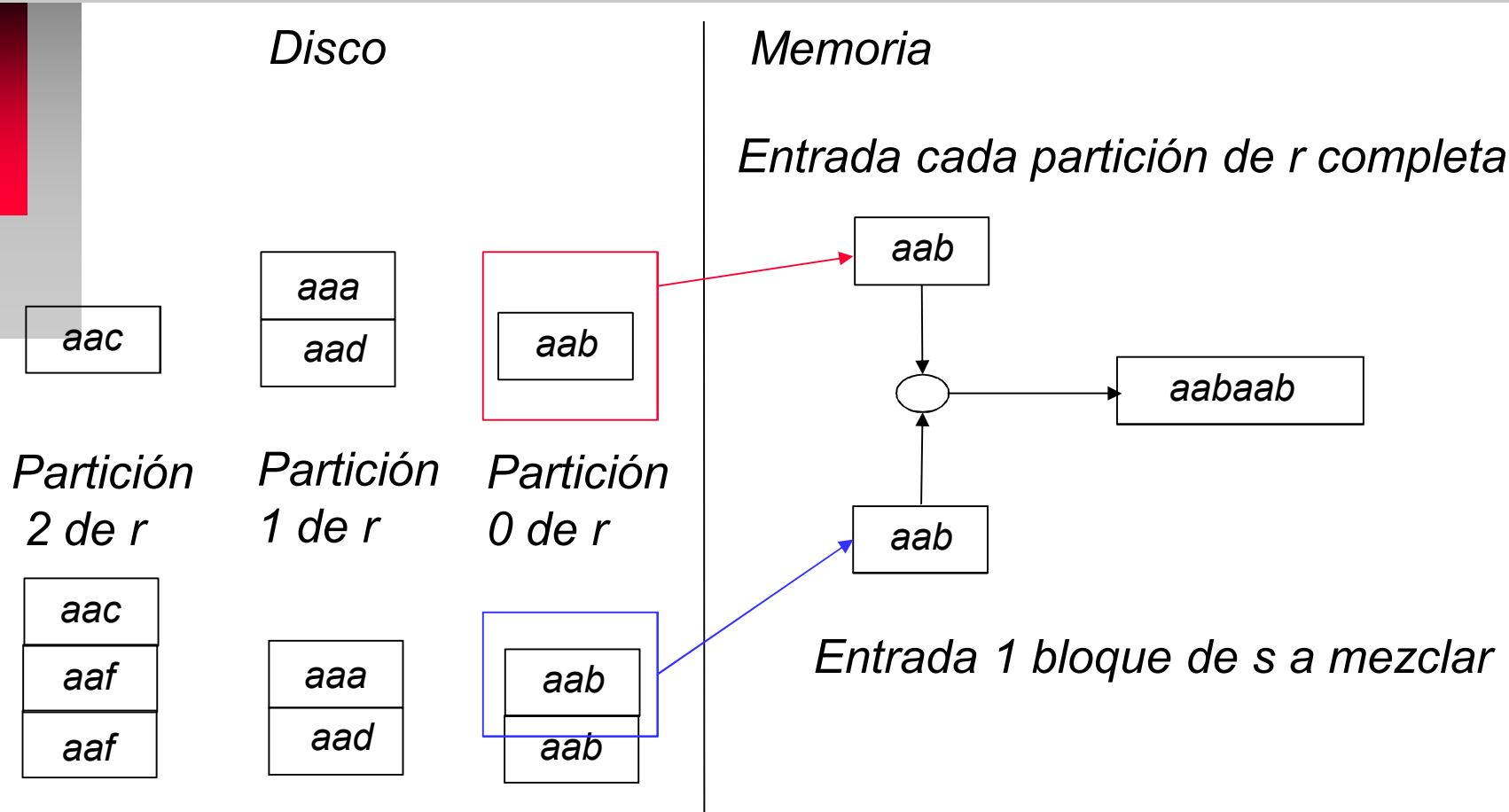
La relación s se llama entrada para construir

y r entrada para probar

Ejemplo de reunión por asociación



Ejemplo de reunión por asociación



Coste de la reunión por asociación: Ejemplo

cliente $\bowtie$ *impositor*

- *Asumir que el tamaño en memoria es de 20 bloques*
- $b_{impositor}$ *= 100 and* $b_{cliente}$ *= 400.*
- *Impositor se usa como relación de construcción. Se particiona 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*

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***
- *Ejplo:. 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* $\Rightarrow$ *usando la primera para probar*
- *Coste* $\Rightarrow$ *3(80 + 320) + 20 +80 = 1300 bloques de transferencia en vez de 1500*
- *Es más útil* $\Rightarrow$ M >> $\sqrt{b_s}$

Reuniones Complejas

- *Reunión con condición de conjunción*

  $r \bowtie_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n} s$

  - *Usar bucle anidado / bucle anidado por bloques ó*
  - *Calcular el resultado de una de las reuniones simples* $r \bowtie_{\theta i} s$
    - *El resultado final consiste en tuplas del resultado intermedio que satisfacen el resto de condiciones*

    $$\theta_1 \wedge \ldots \wedge \theta_{i-1} \wedge \theta_{i+1} \wedge \ldots \wedge \theta_n$$

- *Reunión con condición de disyunción*

  $r \bowtie_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n} s$

  - *Usar bucle anidado / bucle anidado por bloques ó*
  - *Calcular como la unión de los registros de las reuniones* $r \bowtie_{\theta i} s$:

    $(r \bowtie_{\theta 1} s) \cup (r \bowtie_{\theta 2} s) \cup \ldots \cup (r \bowtie_{\theta n} s)$

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 $\Rightarrow$ duplicados estarán en el mismo cajón*
- ***Proyección:*** *se implementa realizando la proyección de cada tupla seguida de la eliminación de los duplicados*

Otras Operaciones: Agregación

- *Se puede implementar de manera similar a la eliminación de duplicados*
  - *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.*

- *Ejemplo: Operaciones sobre conjuntos usando asociación*

  1. *Particionar ambas relaciones usando la misma función de asociación, creando,* r_0 $r_{1, .., }$ r_n, *y* s_0 , s_1, $s_{2..,}$ 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.*

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 ⟕ s
  - *En* r ⟕ s*, las tuplas que no participan están en* r – Π_R(r ⋈ s)
  - *Modificar mezcla* r ⟕ s: *para cada tupla* t_r *de* r *que no cumple con la tuple de* s, *la salida de* t_r *se añade con nulos.*
  - *La runión externa por la derecha y la reunión externa se hace similarmente.*
- *Modificar la reunión por asociación* r ⟕ 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 tupas de r con las de s, al final de* s_i *la salida de las tuplas no coincidentes salen con nulos.*

Ejemplo coste

- *Suponer esta consulta: (r1 ⋈ r2) ⋈ r3 donde*
  - *r1 tiene 10000 tuplas, 1000 bloques*
  - *r2 tiene 20000 tuplas, 2000 bloques*
  - *r3 tiene 30000 tuplas, 3000 bloques*
  - *r1 ⋈ 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?*