Capítulo 3

Operadores sobre tablas y su implementación – parte 2

Ordenamiento

- Especificamos **ordenamiento** usando la idea de insertion sort.
- La función *insert* inserta la tupla *t* en la tabla *r* en la posición que le corresponde: a izquierda las tuplas menores y a derecha las mayores.

$$\mathbf{insert}_{a_1,\ldots,a_N}\ \mathrm{t}\ \mathrm{r} = \sigma_{(a_1,\ldots,a_N)<\mathrm{t}[a_1,\ldots,a_N]}(\mathrm{r}) \mathbin{++} [\mathrm{t}] \mathbin{++} \sigma_{(a_1,\ldots,a_N)\geq\mathrm{t}[a_1,\ldots,a_N]}(\mathrm{r})$$

- El **ordenamiento ascendente** se define:
- $O_{a1,\ldots,aN}([]) = []$
- $O_{a1,\ldots,aN}(x{:}r) = \mathrm{insert}_{a1,\ldots,aN}(x, O_{a1,\ldots,aN}(r))$

Ordenamiento

- El **ordenamiento descendente** se define como la reversa del ordenamiento ascendente.
- $O^{>}_{a1,\ldots,aN}(r) = \text{reverse}\ (O_{a1,\ldots,aN}(r))$
- Reverse [] = []
- Reverse (x : xs) = reverse xs ++ [x]
- Cuando queramos ordenar toda la tabla, omitiremos la lista de las columnas.

Operadores físicos de ordenamiento

- Ya saben ordenar tablas que caben en memoria principal.
- Estudiamos el caso de ordenamiento donde las tablas son mayores que la memoria principal.
  - Ordenar tablas que no caben en memoria se llama **ordenamiento externo**.
  - El algoritmo de ordenamiento externo más usado se llama **ordenamiento-combinación externo (external merge sort)**.

Ordenamiento externo

- Sea *M* la cantidad de bloques en búfer de memoria principal disponibles para ordenación.
  1. **Crear corridas ordenadas**

     i = 0

     repeat

     Leer *M* bloques de la tabla en memoria

     ordenar esos bloques en memoria

     Escribir los datos ordenados al archivo de ejecución R_i,

     i = i+1

     until el fin de la tabla

     - Sea N el valor final de i
  2. **Combinar las corridas**

Ordenamiento externo

- Luego las corridas son combinadas. Suponemos que $N < M$.

  Leer un bloque de cada uno de los *N* archivos Ri en un bloque del búfer de memoria de corrida Ri.

  repeat

  Seleccionar la primera tupla en el orden de ordenamiento entre todos los bloques del búfer.

  Escribir la tupla al búfer de salida y borrarla del bloque del búfer donde estaba.

  if búfer de salida está lleno **then** escribirlo a disco, borrar contenido de búfer de salida.

  if búfer de corrida Ri está vacío y no fin de archivo Ri **then** leer el próximo bloque de la corrida Ri en el bloque de búfer de memoria correspondiente a esa corrida.

  until todos los bloques de búfer de las corridas estén vacíos.

Ordenamiento externo

- Si $N \geq M$, varios ciclos de combinación son requeridos.
  - Se dividen las corridas en grupos contiguos de M-1 corridas.
  - Cada grupo de M-1 corridas es combinado (usando el algoritmo anterior), dando lugar a una nueva corrida.
  - Hacer los dos pasos anteriores constituye una pasada.
    - La misma reduce el número de corridas por un factor de M-1 y crea corridas más largas por el mismo factor.
  - Si el número de corridas resultantes de esa pasada es mayor a M,
    - se realiza otra pasada de la misma manera, pero con las corridas creadas en la primera pasada.
    - Observar que cada pasada reduce el número de secuencias en M-1.
  - Se repiten las pasadas hasta que número de corridas generadas sea menor que *M*.
  - Luego el último ciclo de combinación genera el resultado ordenado.

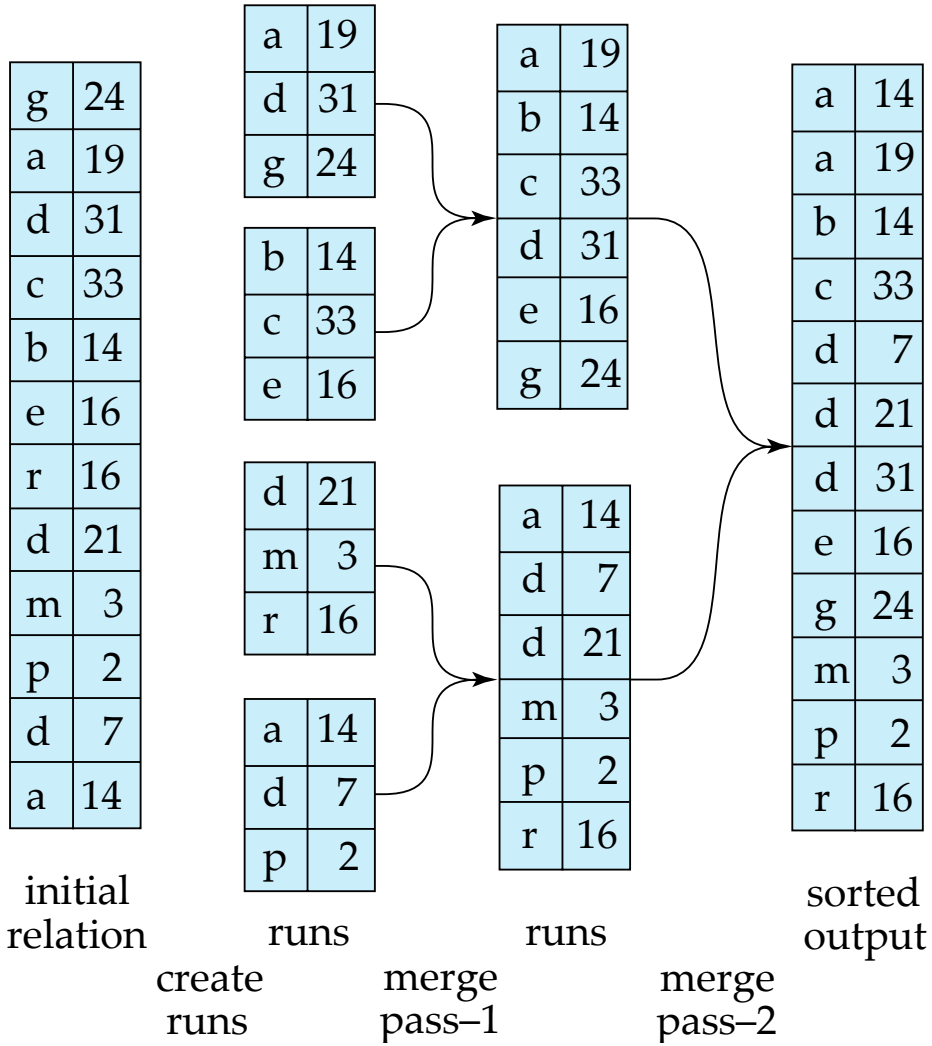
Ordenamiento externo

Ejemplo: Se asume que:

- En memoria entran 3 bloques (M = 3).
- Se asume que bloques son de 1 tupla cada uno.

El algoritmo opera así:

- Primero se crean 4 corridas ordenadas.
- Luego se consideran grupos contiguos de M-1 = 2 corridas.
  - Se combinan grupos contiguos de 2 corridas. Se hacen 2 combinaciones.
  - Se obtienen así 2 corridas.
- Las dos corridas finales son combinadas dando el resultado ordenado final.



Ordenamiento externo

- **Costo del ordenamiento:**
  - Leer justificación en el libro de Silberschatz.
  - Número de transferencias de bloques para ordenamiento externo:

    $b_r (2 \lceil \log_{M-1} (b_r / M) \rceil + 1)$
  - Donde b_r es cantidad de bloques de la tabla
  - Cantidad de accesos a bloque:

    $2 \lceil b_r / M \rceil + \lceil b_r / b_b \rceil (2 \lceil \log_{\lfloor M/bb \rfloor - 1} (b_r / M) \rceil - 1)$,
  - donde b_b bloques son alojados en cada corrida

Reunión selectiva

- Unimos las siguientes tablas mediante el campo *legajo*:

profe			
legajo	nombres	apellidos	sueldo
p1	"Benjamin"	"Pierce"	3000
p2	"Patricia"	"Selinger"	6000
p3	"Edgar F"	"Codd"	5500
p4	"Barbara"	"Liskov"	5600

curso		
id	legajo	nombre
c1	p2	"Optimización de Consultas"
c2	p3	"Fundamentos de BD"
c3	p2	"Análisis de Datos"
c4	p1	"Fundamentos del Software"
c5	p4	"Programación OO"

_						
legajo	nombres	apellidos	sueldo	id	legajo	nombre
p1	"Benjamin"	"Pierce"	3000	c4	p1	"Fundamentos del Software"
p2	"Patricia"	"Selinger"	6000	c1	p2	"Optimización de Consultas"
p2	"Patricia"	"Selinger"	6000	c3	p2	"Análisis de Datos"
p3	"Edgar F"	"Codd"	5500	c2	p3	"Fundamentos de BD"
p4	"Barbara"	"Liskov"	5600	c5	p4	"Programación OO"

- **La operación de combinar tablas por atributos en común es muy utilizada.** $\text{profe} \underset{\text{legajo}}{\bowtie} \underset{\text{legajo}}{\text{curso}}$

Reunión selectiva

- Ahora pasamos a ver el esquema de la reunión selectiva.
- Sean dos tablas de esquema:

$$r(n_1 :: \tau_1, \ldots, n_N :: \tau_N) \text{ y } s(m_1 :: \tau'_1, \ldots, m_M :: \tau'_M)$$

- Entonces:

$$r \underset{a_1,\ldots,a_i}{\bowtie}_{b_1,\ldots,b_i} s :: (n_1 :: \tau_1, \ldots, n_N :: \tau_N, c_1, \ldots, c_{M-i})$$

$$\text{para } j \in [1, M-i],\ c_j \in \{m_i, \ldots, m_M\} \setminus \{b_1, \ldots, b_i\}.$$

Reunión selectiva

- La reunión selectiva se puede definir usando los operadores vistos:

$$r \ {}_{a_1,\ldots,a_i}\bowtie_{b_1,\ldots,b_i} s = \Pi_{n_1,\ldots,n_N,c_1,\ldots,c_{M-i}}(\sigma_{a_1=b_1 \wedge \ldots \wedge a_i=b_i}(r \times s))$$

- Un tipo de reunión selectiva que se usa mucho es **reunión natural**: se aplica reunión selectiva a todos los atributos con el mismo nombre en las dos tablas.
- Sean $r(n_1 :: \tau_1, \ldots, n_N :: \tau_N)$ y $s(m_1 :: \tau'_1, \ldots, m_M :: \tau'_M)$, definimos la reunión natural de *r* y *s* como:

$$r \bowtie s = r \ {}_{a_1,\ldots,a_i}\bowtie_{a_1,\ldots,a_i} s$$

donde $\{a_1, \ldots, a_i\} = \{n_1, \ldots, n_N\} \cap \{m_1, \ldots, m_M\}$.

Reunión natural

- Relaciones *r*, *s*:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

- Los atributos en común son: B y D
- El esquema de la reunión natural de r y s: (A, B, C, D, E)

- $r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Operadores físicos para reunión selectiva

- **Algoritmo de reunión selectiva de loop anidado**
- **for each** tuple t_r **in** r **do begin**
  for each tuple t_s **in** s **do begin**
  test pair (t_r, t_s) to see if they satisfy the join condition
  if they do, add $t_r \bullet t_s$ to the result.
  end
  end

Operadores físicos para reunión selectiva

- Nomenclatura: *r* se llama **tabla externa** y *s* se llama **tabla interna** de la reunión selectiva.
- El algoritmo anterior no requiere de índices.
- El algoritmo anterior es costoso porque examina todo par de tuplas en las dos tablas.
- En el peor caso, si hay suficiente memoria solo para sostener un bloque para cada tabla, el costo estimado es:
  - $n_r * b_s + b_r$ transferencias de bloques
  - $n_r + b_r$ accesos a bloques.
  - donde n_r es cantidad de registros en *r* , b_r es la cantidad de bloques en *r,* y b_s es la cantidad de bloques en *s.*

Operadores físicos para reunión selectiva

- **Algoritmo de reunión selectiva de loop anidado de bloques**

for each block B_r **of** r **do begin**
for each block B_s **of** s **do begin**
for each tuple t_r **in** B_r **do begin**
for each tuple t_s **in** B_s **do begin**
if (t_r, t_s) satisface condición de reunión selectiva
then add $t_r \bullet t_s$ to the result.
end
end
end
end

Operadores físicos para reunión selectiva

- **Estimación de peor caso:**
  - $b_r * b_s + b_r$ transferencias de bloque + 2 * b_r accesos a bloque
  - Cada bloque en la tabla interna *s* es leído una sola vez por cada bloque en la tabla externa.
- **Mejora del algoritmo anterior:**
  - M tamaño de memoria en bloques.
  - Usar M-2 bloques de disco para tabla externa, usar los restantes 2 bloques para tabla interna y salida.
  - Costo = $\lceil b_r / (M-2) \rceil * b_s + b_r$ transferencias de bloques + $2\lceil b_r / (M-2) \rceil$ accesos a bloque

Operadores físicos para reunión selectiva

- **Algoritmo de reunión selectiva de loop anidado indexado**
  - Asumir que hay un índice en el atributo de la relación interna de la reunión.
  - Para cada tupla t_r en la tabla externa *r* usar el índice para buscar tuplas en *s* que satisfacen la condición de reunión con tupla t_r.
  - **Peor caso**: el búfer tiene espacio para solo un bloque de *r* y un bloque del índice y para cada tupla en *r* hacemos búsqueda en índice de *s*.
    - Para leer *r* en el peor caso hay b_r accesos a bloque y b_r transferencias de bloque.
    - A esto hay que sumarle $n_r * c$. n_r es cantidad de registros en *r.*
    - Aquí *c* es el costo de recorrer el índice y recolectar todas las tuplas de *s* que cazan para una tupla de *r*.
    - El valor *c* puede estimarse como el costo de una selección en *s* usando la condición de la reunión.

Operadores físicos para reunión selectiva

Algoritmo de reunión por mezcla:

- Supongamos que hay un atributo de la reunión.
- Ordenar ambas tablas en el atributo de la reunión.
- Aplicar la reunión selectiva a las tablas ordenadas.
  - Si las tablas consideradas son r y s y se quiere hacer reunión selectiva de r y s:
  - Se trabaja con variables para tupla actual de r y tupla actual de s
  - El algoritmo se basa en aplicar las siguientes dos ideas:
    - Si tupla actual t de r caza con tupla actual t' de s: se agrega t ⋈ t' al resultado.
      - Se repite esto hasta que haya tupla de s que no caza con t o se acabó de recorrer s.
    - Si tupla actual de r no caza con tupla actual de s, se pasa a siguiente tupla de r
      - Se repite esto hasta que se acabe de recorrer r o se encuentra tupla de r que caza con tupla actual de s.

Operadores físicos para reunión selectiva

- **Análisis del algoritmo de reunión por mezcla:**
  - Cada bloque necesita ser leído una sola vez asumiendo que todas las tuplas para un valor dado de los atributos del join entran en memoria.
  - Entonces el costo de reunión por mezcla es:
    - $b_r + b_s$ transferencia de bloques $+\lceil b_r / b_b \rceil + \lceil b_s / b_b \rceil$ accesos a bloque
    - Donde b_b bloques de búfer son asignados a cada tabla.
    - A esto se suma el costo de ordenar las tablas si no estaban ordenadas.

Operadores físicos para reunión selectiva

- **Algoritmo de reunión por mezcla híbrido:**
- **Suposición**: la primera tabla está ordenada y la segunda tabla está desordenada, pero tiene un índice secundario árbol B+ en los atributos de la reunión.
- **Algoritmo**:
  1. El algoritmo combina la tabla ordenada con las entradas hoja del índice secundario B+.
     - El archivo resultante contiene tuplas de la relación ordenada y direcciones para las tuplas de la relación no ordenada.
  2. Se ordena el resultado por las direcciones de las tuplas de la relación no ordenada,
  3. Permitiendo el retorno eficiente de las tuplas correspondientes en el orden de almacenamiento físico para terminar la reunión.
- Costo del segundo paso: Costo de ordenar el archivo resultante.
- Costo del tercer paso: costo de acceso a tuplas en el orden de almacenamiento físico.

Renombre

- Sea la consulta: obtener el legajo de los profesores con sueldo anual > 60000.
- Tenemos de antes:

$$\text{sueldos\_anuales} = \Pi_{\text{legajo},\ \text{sueldo}*13}(\text{profe})$$

- No podemos usar esta consulta y aplicarle selección.
- Para eso necesitamos darle nombre a la segunda columna de esta tabla.
- Esto se puede hacer con un operador de renombramiento más general que el anterior.

Renombre

- Sea la tabla:

$$\text{Sea } r(n_1 :: \tau_1, \ldots, n_N :: \tau_N).$$
$$\rho_{s(n'_1,\ldots,n'_N)}(r) :: s(n'_1 :: \tau_1, \ldots, n'_N :: \tau_N)$$
$$\rho_{s(n'_1,\ldots,n'_N)}(r) = r$$

- No lo ponemos, pero en el esquema de r algunos campos pueden no tener nombre.
- Entonces obtener el numero de legajo de profes con sueldo anual > 60000:
- $\sigma_{\text{sueldoAnual} > 600000} (\rho_{\text{profe2(nro\_legajo, sueldAnual)}} (\Pi_{\text{nro\_legajo, sueldo * 13}} (\text{profe})))$

Remover duplicados

- La operación de remoción de duplicados se define recursivamente así:

1. V [ ] = [ ]
2. V (t:r) = if t∈r then V (r) else t: V (r)

- El predicado en la segunda ecuación se define del siguiente modo:
- $t \in [\,] = \text{false}$
- $t \in (u : r) = t == u \,||\, t \in r$

Operador físico de remoción de duplicados

- **Implementación de remoción de duplicados**
  - Se ordena la tabla (según todos los atributos);
    - al hacerlo, todos los duplicados van a ser adyacentes entre sí.
    - Y todos los duplicados menos uno, pueden ser eliminados.
  - **Optimización**:
    - en ordenación externa duplicados pueden ser eliminados durante generación de corridas,
    - así como en pasos de combinación intermedios.
    - El peor caso de costo para eliminación de duplicados es el mismo que el peor caso de costo para ordenar una tabla.

Concatenación de tablas

- ¿Qué operación de tablas corresponde a la unión de conjuntos?
- Una posibilidad es la **concatenación de tablas**.
- Es como la concatenación de listas solo que ahora el resultado tiene un esquema.
- Los esquemas de los operandos deben ser **compatibles (**i.e. igual cantidad de columnas y mismos tipos en las mismas posiciones**)**.

$$\text{Sean } r(n_1 :: \tau_1, \ldots, n_N :: \tau_N) \text{ y } s(m_1 :: \tau_1, \ldots, m_M :: \tau_N)$$

$$r{+}{+}s :: (n_1 :: \tau_1, \ldots, n_N :: \tau_N)$$

- [] ++ s = s
- (t:r)++ s = t: (r++s)

Concatenación

- Concatenación requiere leer ambas tablas: primero la de la izquierda y luego la de la derecha.
  - A medida que se lee una tabla se genera el resultado.
- **Análisis de costo**:
  - En el peor caso para computar r++s vamos a necesitar:
    - $b_r + b_s$ transferencias de bloques y
    - $b_r + b_s$ accesos a bloques (cuando todos los bloques de *r* y *s* no están contiguos)
- **Nota**: Si el resultado es un resultado intermedio: se escriben $b_r + b_s$ bloques en disco.

Resta

Los esquemas de las tablas deben ser compatibles

$$r(n_1 :: \tau_1, \ldots, n_N :: \tau_N) \text{ y } s(m_1 :: \tau_1, \ldots, m_M :: \tau_N)$$

El esquema de la resta es el del primer operando.

$$r \setminus s :: (n_1 :: \tau_1, \ldots, n_N :: \tau_N)$$

$$r \setminus s = \sigma_{(\backslash \mathsf{t} \rightarrow \mathsf{t} \notin \mathsf{s})}(r)$$

Resta

- **Algoritmo**:
  - Para calcular la resta de dos tablas, podemos primero **ordenarlas** a ambas por la clave primaria.
  - Luego podemos escanear una vez cada tabla ordenada para producir el resultado.
- **Costo**:
  - Suponiendo las tablas r, s ordenadas de ese modo.
  - Si un solo bloque en búfer se usa por tabla:
    - $b_r + b_s$ transferencias de bloques
    - $b_r + b_s$ accesos a bloque en el peor caso (por ejemplo, cuando r = s)

Intersección

- Los esquemas de las tablas deben ser compatibles

$$r(n_1 :: \tau_1, \ldots, n_N :: \tau_N) \text{ y } s(m_1 :: \tau_1, \ldots, m_M :: \tau_N)$$

- El esquema de la intersección es el del primer operando.

$$r \cap s :: (n_1 :: \tau_1, \ldots, n_N :: \tau_N)$$

$$r \cap s = \sigma_{(\mathsf{t} \rightarrow \mathsf{t} \in \mathsf{s})}(r)$$

intersección

- **Algoritmo**:
  - Para calcular la intersección de dos tablas, podemos primero **ordenarlas** a ambas por la clave primaria.
  - Luego podemos escanear una vez cada tabla ordenada para producir el resultado.
- **Costo**: Suponiendo las tablas r, s ordenadas de ese modo, $r \cap s$ requiere:
  - Si un solo bloque en búfer se usa por tabla:
    - $b_r + b_s$ transferencias de bloques
    - $b_r + b_s$ accesos a bloque en el peor caso
  - El número de accesos a bloque puede ser reducido alojando bloques extra en búfer.