Capítulo 4 - Factor de selectividad y operadores físicos

Factor de selectividad

Consideramos:

- $fs(P, r)$ para propiedad P y tabla r
- A, B atributos de r
- c, d constantes
- $V(A, r)$ es el número de distintos valores que aparecen en r para A

Luego, las **reglas** del factor de selectividad son (recordar que se asume *uniformidad* e *independencia*):

- $fs(A = c, r) = \frac{1}{V(A,r)}$
- Con A con valor numérico
  - $fs(A \geq c, r) = \frac{max(A,r) - c}{max(A,r) - min(A,r)}$
  - $fs(A < c, r) = \frac{c - min(A,r)}{max(A,r) - min(A,r) + 1}$
  - $fs(c \leq A < d) = \frac{d - c}{max(A,r) - min(A,r)}$
- $fs(P_1 \wedge \ldots \wedge P_n) = \prod_{i=1}^{n} fs(P_i, r)$
- $fs(\neg P, r) = 1 - fs(P, r)$
- $fs(P \vee Q, r) = 1 - [(1 - fs(P, r)) \times (1 - fs(Q, r))]$

$$
\begin{aligned}
fs(P \vee Q, r) = \\
= fs(\neg(\neg(P \vee Q)), r) \\
= fs(\neg(\neg P \wedge \neg Q), r) \\
= 1 - fs(\neg P \wedge \neg Q, r) \\
= 1 - [fs(\neg P, r) \times fs(\neg Q, r)] \\
= 1 - [(1 - fs(P, r)) \times (1 - fs(Q, r))]
\end{aligned}
$$

Operadores físicos

Consideraciones para los costos

- $b_r \rightarrow$ número de bloques conteniendo registros de la tabla r
- $t_S \rightarrow$ tiempo de búsqueda requerido
- $t_T \rightarrow$ tiempo de transferencia de cada bloque
- $h_i \rightarrow$ altura del índice i en el árbol B^+

Selección

Algoritmo de búsqueda lineal

- Proceso
  - Escanear cada bloque del archivo
  - Para cada bloque, testear todos los registros para ver si satisfacen la condición
  - Quedarnos con los registros que la cumplen
- Costos
  1. b_r (transferencias de bloques) $+ 1$ (acceso a bloque)
  2. $t_s + b_r \times t_r$
- Para **igualdad de clave**, el costo *promedio* es con $\frac{b_r}{2}$ en vez de b_r

Algoritmo para índice primario usando árbol B^+

- **Igualdad de clave**
  - *Proceso*
    - Se recorre la altura del árbol más una E/S para recoger el registro
    - Cada una de las operaciones requiere un acceso y transferencia de bloque
  - *Costo*
    - $(h_i + 1) \times (t_T + t_S)$
- **Igualdad de atributo no clave**
  - *Proceso*
    - Hay un acceso a bloque para cada nivel del árbol y uno más para el primer bloque
    - Consideramos b como el número de bloques conteniendo valores que cumplen, y asumiendo que están almacenados secuencialmente y no requieren accesos adicionales a bloque
  - *Costo*
    - $h_i \times (t_T + t_S) + t_T \times b + t_S$
- **Comparaciones**
  - $\sigma_{A \geq v}(r)$
    - Usar el índice para encontrar el primer registro $\geq v$ y escanear la tabla secuencialmente desde allí
    - Sea b el número de bloques conteniendo registros $A \geq v$, el *costo* es de $h_i \times (t_T + t_S) + b \times t_T$
  - $\sigma_{A \leq v}(r)$
    - No se usa el índice
    - Se escanea la tabla secuencialmente hasta la primer tupla $> v$

Algoritmo para índice secundario usando árbol B^+

- **Igualdad de clave**
  - *Proceso*
    - Similar al de índice primario
  - *Costo*
    - $(h_i + 1) \times (t_T + t_S)$

- **Igualdad de atributo no clave**
  - *Proceso*
    - Similar al de índice primario pero aquí cada registro puede estar en un bloque diferente, requiriendo un acceso a bloque por registro
    - Consideramos n como el número de registros requeridos
  - *Costo*
    - $(h_i + n) \times (t_T + t_S)$
    - Puede ser **muy caro**
- **Comparaciones**
  - $\sigma_{A \geq v}(r)$
    - Usar el índice para encontrar la primer entrada del índice $\geq v$ y escanear el índice secuencialmente desde allí para encontrar los punteros a los registros
    - Sea n el número de registros con $A \geq v$, el *costo* es de $(h_i + n) \times (t_T + t_S)$
  - $\sigma_{A \leq v}(r)$
    - Escanea hojas del índice encontrando punteros a los registros, hasta la primer entrada $> v$
  - Como se requiere en *ambos casos* una E/S para cada registro, la *búsqueda lineal* en la tabla puede ser más barata !!!

Proyección (y proyección generalizada)

- Proceso
  - Recorre todos los registros
  - Realiza una proyección en cada uno
- Costo
  - b_r (transferencias de bloques) $+ 1$ (acceso a bloque)

Ordenamiento

- *Consideraciones*
  - Tomamos el caso de ordenamiento donde las tablas son mayores que la memoria principal
  - Este tipo de ordenamiento se llama **ordenamiento externo**
    - El más usado es el **ordenamiento-combinación externo (external merge sort)**
  - Consideramos que M es la cantidad de bloques en búfer de memoria principal disponibles para ordenación
- *Proceso*
  - Fase 1: crear corridas ordenadas
    - Para cada "pedazo" de M bloques de la tabla, lo que hacemos es leerlos, ordenarlos y escribir los datos ordenados en el archivo de ejecución R_i (el que contiene el valor de mi corrida)
    - Consideramos que N es el valor final de cantidad de corridas
  - Fase 2: combinar corridas
    - Si $N < M$

        - Leemos un bloque de cada uno de los N archivos de corrida y los ponemos en un bloque del búfer de memoria
        - Luego, vamos seleccionando el más chico y quedándonos con ese
        - Actualizamos para considerar el siguiente registro de ese bloque
            - En caso que lo hayamos completado, pedimos leer el próximo bloque de esa corrida
        - Mientras tanto, los resultados van siendo escritos a disco (cuando el búfer de salida esté lleno)
        - Apenas no nos queden bloques en el búfer de entrada, terminamos el algoritmo
    - Si $N \geq M$
        - Son requeridas **varias** pasadas de combinación
        - En cada una, grupos continuos de $M-1$ corridas son combinados, por lo que se reduce el número de corridas por un factor de $M-1$ pero se aumenta su tamaño por lo mismo
        - Esto es repetido hasta que todas las corridas hayan sido combinadas en una sola
- *Costo*
    - Número de transferencias de bloques (b_r es cnt de bloques de la tabla)
        - $b_r \times (2 \times \lceil \log_{M-1}(\frac{b_r}{M}) \rceil + 1)$
    - Cantidad de accesos a bloques (b_b es cnt de bloques alojados en cada corrida)
        - $2 \times \lceil \frac{b_r}{M} \rceil + \lceil \frac{b_r}{b_b} \rceil \times (2 \times \lceil \log_{\lfloor \frac{M}{b_b} \rfloor - 1}(\frac{b_r}{M}) \rceil - 1)$

Reunión selectiva (y natural)

- *Consideraciones*
    - Tenemos tablas r, s (si se itera, se hace primero sobre r)
    - n_r, n_s son cnt. de registros en r y s respectivamente
    - b_r, b_s son cnt. de bloques en r y s respectivamente
- **Loop anidado**
    - *Proceso*
        - Ver todos los pares de tuplas posibles y si se satisface la condición, agregar su combinación
        - No requiere de índices pero es costoso porque examina todo par de tuplas en las dos tablas
    - *Costo*
        - $n_r \times b_s + b_r$ transferencias de bloques
        - $n_r + b_r$ accesos a bloques
- **Loop anidado de bloques**
    - *Proceso*
        - Ver todos los pares de bloques posibles y, dentro de cada uno de estos, iterar por sus pares de tuplas
        - No requiere índices y se asegura de que los registros que se accedan para una combinación de bloques no requieran otra transferencia de bloques
    - *Costo*

        - $b_r \times b_s + b_r$ transferencias de bloque
        - $2 \times b_r$ accesos a bloque
            - Cada bloque en la tabla interna s es leído una sola vez por cada bloque en la tabla externa
    - *Mejora*
        - Sea M el tamaño de la memoria en bloques, se usan $M - 2$ bloques para tabla externa y los 2 restantes para tabla interna y salida
        - El *costo* pasa a ser de
            - $\lceil \frac{b_r}{M-2} \rceil \times b_s + b_r$ transferencias de bloques
            - $2 \times \lceil \frac{b_r}{M-2} \rceil$ accesos a bloque
- **Loop anidado indexado**
    - *Proceso*
        - Mismo algoritmo que el *loop anidado*, pero se usa un índice en el atributo de la relación interna de la reunión (para buscar las tuplas en s)
    - *Costo*
        - Se considera el peor caso, cuando 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
        - $b_r \times (t_T + t_S) + n_r \times c$
            - c es el costo de recorrer el índice y recolectar todas las tuplas de s que cazan para una tupla de r
            - c puede estimarse como el costo de una selección en s usando la condición de la reunión
- **Join merge-sort**
    - *Proceso*
        - Se ordenan ambas tablas según el atributo de la reunión
        - Se hace merge de las relaciones ordenadas para hacer la reunión
            - En el caso de valores duplicados, cada par con el mismo valor de atributo del join debe ser *juntado*
        - 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
    - *Costo*
        - $b_r + b_s$ transferencia de bloques
        - $\lceil \frac{b_r}{b_b} \rceil + \lceil \frac{b_s}{b_b} \rceil$ accesos a bloques
            - b_b bloques de búfer son asignados a cada tabla
        -
            - costo de ordenar si las tablas no están ordenadas

Agregación

- *Proceso*
    - Se usa ordenación para juntar tuplas del mismo grupo y luego las funciones de agregación pueden ser aplicadas a cada grupo
    - *Truquitos de optimización*

        - En lugar de primero agrupar las tuplas por grupo y luego aplicar los operadores de agregación, se puede implementar para que estos sean calculados a medida que se construyen los grupos
            - Para el caso de sum/min/max, cuando dos tuplas del mismo grupo son encontradas, se reemplaza por una sola conteniendo el valor necesitado
            - Para count, se mantiene un count de cada grupo para el cual una tupla ha sido encontrada
            - Para avg., se computan sum y count, y luego se hace $\frac{sum}{count}$ al final
- *Costo*
    - Es el mismo que el costo de ordenar una tabla

Concatenación

- *Proceso*
    - Se leen ambas tablas, primero la de la izquierda y luego la de la derecha
    - Con ello se va generando la tabla resultado
- *Costo* (peor caso)
    - $b_r + b_s$ transferencias de bloques y accesos a bloques (i.e., $2 \times (b_r + b_s)$)
    - Si el resultado es *intermedio*, se escriben $b_r + b_s$ bloques en disco

Intersección

- *Proceso*
    - Se ordenan ambas tablas por PK
    - Luego se escanea una vez cada tabla ordenada para producir el resultado
- *Costo*
    - $b_r + b_s$ transferencias de bloques y accesos a bloques (i.e., $2 \times (b_r + b_s)$)
    - El número de acceso a bloque puede ser reducido alojando bloques extra en búfer de memoria principal

Resta

- *Procedimiento*
    - Se ordenan ambas tablas por PK
    - Luego se escanea una vez cada tabla ordenada para producir el resultado
- *Costo*
    - $b_r + b_s$ transferencias de bloques y accesos a bloques (i.e., $2 \times (b_r + b_s)$)