CC42A – BASES DE DATOS
Profesores: Claudio Gutiérrez, Gonzalo Navarro
Auxiliar: Mauricio Monsalve

Auxiliar 7

- Simulación de control

Simulación de control

A continuación hay cuatro preguntas con dificultad de control. Para ser más precisos, estas preguntas son leves variaciones a preguntas de libros y de controles anteriores. Tómese su tiempo para responder y pensar. Revise los apuntes tantas veces como quiera.

Problema 1 (1,5 puntos)

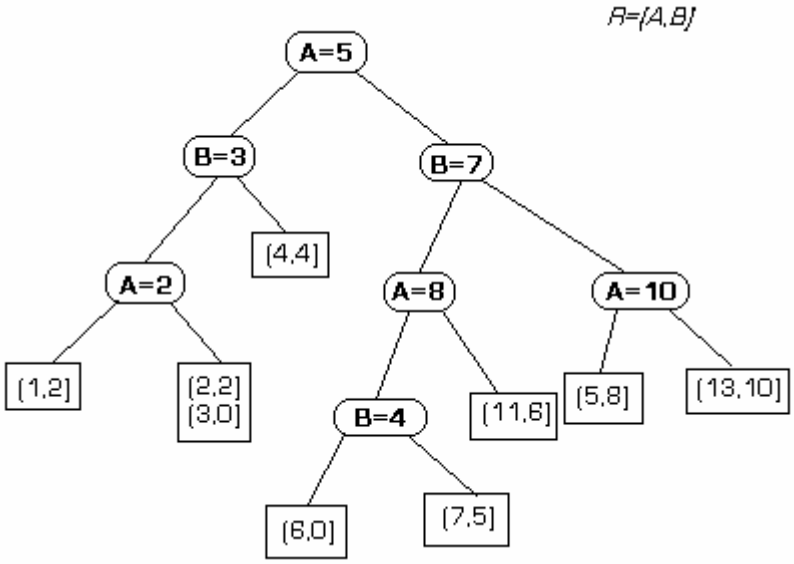
Se define el operador relacional semijoin de la siguiente forma: *A semijoin B* = $\pi_{Atribs(A)}(A * B)$. Proponer algoritmos para resolución de esta consulta en los siguientes escenarios: Las tablas A y B están ordenadas, están desordenadas, tienen índices árbol B^+ y tienen índices hash. Para cada caso indicar el costo.

Problema 2 (1 punto)

Proponga un algoritmo para una consulta SELECT … GROUP BY *f(X)* para los siguientes escenarios: Tablas ordenadas, tablas desordenadas, índices árbol B^+ e índices hash. Indicar alguna diferencia en casos cuando hay, o no, HAVING *g(X)*.

Problema 3 (1,5 puntos)

Un árbol kd (*kd-tree*) es un tipo de árbol multidimensional basado en un árbol binario de búsqueda pero de nodos alternantes, como se ve a continuación:



Suponga que se hace una consulta de la forma SELECT … WHERE *X* OR *Y* y otra de la forma SELECT … WHERE *X* AND *Y*. Indique cómo se resolverían estas consultas en el *kd-tree*.

Problema 4 (2 puntos)

Proponga estrategias para la división ($A \div B$) en los casos sin índices y con índices (B^+ y hash).

Propuesta de pauta

Aquí se esbozará, en líneas generales, cómo serían las respuestas a las distintas preguntas:

P1.- Entre que sean tablas sin índices y con índices hay diferencia. Típicamente una tabla desordenada sin índices se ordena, sin embargo se evalúa la posibilidad de agregar índices. En el caso con índices la intuición va por tomar un bloque de A y ver si hay uno o más bloques de B que pueden ser asignados a A por join natural. Si esto es posible, entregar la fila de A, ignorar de lo contrario. Y seguir así. Ojo que en el caso de hashing el proceso anterior es muy rápido, basta analizar la función de hashing.

P2.- Esta es para ver ejemplos de cómo aplicar técnicas de diseño de algoritmos en BD:

- Si no hay HAVING:

Se pueden agrupar los atributos, acumular un valor para cada grupo y entregar el resultado.

Usar hash en atributos agrupados. Examinar buckets y agregar para cada grupo. Ojo con límites de memoria.

Usar índices prehechos en atributos agrupados. Utilizar índices para retornar rápidamente los grupos.

- Si hay HAVING:

En este caso, y luego de verificar que el HAVING es real (no un where), entonces acumular resultados y probar la condición de HAVING. Ojo que usar funciones como el promedio es lo mismo que la suma y el count, operaciones que ambas utilizan nada más que poder de procesador y, en costos I/O, son lo mismo que leer buckets.

P3.- En este caso la intención es entender una nueva estructura de datos, el *kd-tree*. Pero es sólo para probar habilidades de adaptación a nuevos escenarios.

La implementación de la selección (σ) es análoga a la implementación en el caso de árboles B, sólo que teniendo la precaución que en este tipo de árbol hay búsqueda binaria y varios tipos de nodos.

P4.- $A \div B$ puede verse muy difícil, pero se puede abordar de una forma más sencilla. La idea clave es escribir la división como una expresión relacional ($A \div B = \pi_{A'-B'} A - \pi_{A'-B'}\left(\pi_{A'-B'}(A) \times B - A\right)$) y atacar el problema desde ese punto de vista. Sin embargo, con mayor eficiencia, se pueden usar propiedades de índices hash (si es que existen o hacerlos) y analizar si hay elementos en $\pi_{A'-B'}(A)$ que estén relacionados con todos los elementos de B dentro de A.

* Nota: $\pi_{A'-B'} = \pi_{Esq(A)-Esq(B)} = \pi_{Atribs(A)-Atribs(B)}$ ← *Notación SÓLO por comodidad*