Guía de almacenamiento de datos y evaluación de consultas

Mauricio Monsalve M.

June 10, 2007

1 Almacenamiento e índices

1.1 Problemas

1. Indique las características de cada tipo de organización de archivos:
   (a) Desordenados (heap).
   (b) Ordenados (sorted).
   (c) Indexados (indexed).
2. ¿Qué es un índice? ¿Para qué sirve?
3. ¿Cuál es la diferencia entre un índice primario y un índice secundario?
4. ¿Cuál es la diferencia entre un índice denso y un índice disperso?
5. ¿Cuál es la diferencia entre un índice agrupado y uno no agrupado?
6. Indique en qué casos el uso de un índice hashing es peor que otra alternativa:
   (a) Selección de todas las tuplas (scan).
   (b) Búsqueda por igualdad.
   (c) Búsqueda por rango.
7. ¿Por qué una función de hashing debe cumplir con los requerimientos de *pseudoaleatoriedad* y *distribución uniforme*?
8. ¿Cuándo un índice B+ es mejor que un índice hashing?
9. ¿Por qué un índice B+ es mejor que un archivo ordenado?
10. Para las siguientes consultas, índique qué atributo debiera tener un índice y de qué tipo:
    (a) SELECT * FROM A WHERE A.X < 5;
    (b) SELECT * FROM A WHERE A.Y <> 17;
    (c) SELECT * FROM A WHERE A.X = 4 AND A.Y < 10;
    (d) SELECT * FROM A WHERE A.X = -1 OR A.X > 5;
    (e) SELECT * FROM A WHERE A.X < 5 AND A.Y <> 1;
11. Usando los criterios de *pseudoaleatoriedad* y *distribución uniforme*, indique cuáles de las siguientes funciones podrían ser usadas como funciones de hashing para una cadena de texto:

(a) $h_1(x)$ = El valor numérico de la primera letra de la cadena.

(b) $h_2(x)$ = La suma de los valores numéricos de las letras de la cadena.

(c) $h_3(x) = \frac{h_2(x)}{h_1(x)}$.

(d) $h_4(x) = h_2(x)\%100$(operación resto).

12. Construya un árbol B+, con capacidad para 3 punteros, con los siguientes valores: 1, 50, 3, 10, 2, 27, 14, 11, 23.

13. ¿Qué función tiene el índice en A.X en la consulta SELECT * FROM A WHERE A.X LIKE '%.uch.%';?

14. ¿Siempre los índices hacen más rápidas las evaluaciones? Sea R(A,B,C), con un índice agrupado (clustered) sobre A y uno desagrupado sobre B. Entregue un ejemplo de una consulta que se haga más lenta al usar índices, una que se haga más rápida y otra que no se vea afectada.

15. Un R-Tree (árbol R) es un tipo de árbol especializado en la separación de datos multidimensionales. ¿En qué casos usaría un R-Tree como índice?

16. Un índice BITMAP es un índice que consiste en un vector de valores binarios que contienen la respuesta a una condición lógica (participante en el WHERE de alguna consulta) para cada tupla en una tabla. Por ejemplo, el índice bitmap de la relación R(A,B)={(a,0),(b,1),(c,0)} evaluado con B<1 entregaría el bitmap 101 (donde 1 indica el valor verdadero). Responda:

    (a) En una tabla que ocupa 1 Gb en disco y cuyas tuplas ocupan 2 Kb, ¿cuánto espacio en disco ocupa el índice bitmap? Considere sólo el almacenamiento de los valores de verdad.

    (b) ¿Cuáles son los inconvenientes de usar índices de este tipo? ¿En qué casos usaría un índice bitmap?

1.2 Respuestas selectas

1. —

2. —

3. Un índice se dice primario si su clave de búsqueda contiene la llave primaria. De otra manera, se dice secundario.

4. El índice denso apunta al registro. El índice disperso apunta al bloque.

5. —

6. Sólo (c). Sería tentativo incluir (a), pero revisar todas las tuplas es un peor caso que ninguna alternativa puede mejorar.

7. La pseudoaleatoriedad es para *tratar* de asegurar que las claves de búsqueda no estén asociadas a una pequeña porción de la tabla de hashing. La distribución uniforme es para *tratar* de equidistribuir las claves de búsqueda entre los distintos buckets.

8. —

9. Porque B+ rinde mucho mejor en inserciones y eliminaciones que un archivo ordenado.

10. Respuestas elegidas:

    (a) —

    (b) Ninguno en especial.

(c) Si A.X = 4 es más restrictivo que A.Y < 10, entonces conviene hashing. De otra forma, convendría B+ sobre A.Y (y A.X, a la vez).

(d) B+.

(e) —

11. Respuestas elegidas:

    (a) Las primeras letras no se distribuyen uniformemente (por ejemplo, W, X e Y se usan muy poco).

    (b) No tiene cota clara.

    (c) —

    (d) Potencialmente. La operación resto se ocupa muchas veces para generar variables pseudoaleatorias.

12. —

13. B+ y hashing son inútiles en esa situación. Ni un orden por prefijos ni un orden por sufijos ayudaría a B+. Hashing sólo sirve con igualdades.

14. —

15. R-Tree es muy útil en condiciones multiatributo, sean de rango o de igualdad. En condiciones sobre varias columnas y datos geográficos puede ser bastante útil.

16. —

2 Optimización heurística de consultas

2.1 Problemas

1. ¿Cuál es la función del catálogo en la base de datos?

2. Dibuje los árboles de evaluación de las siguientes consultas (usando el protocolo left-deep join) y aplique la heurística de optimización vista en clases:

    (a) SELECT A.X, A.Y FROM A WHERE A.X=A.Y;

    (b) SELECT A.X FROM A,B WHERE A.X=0 AND A.Y=B.Y;

    (c) SELECT A.X,B.Z FROM A,B,C WHERE A.X=B.X AND A.Y=C.Y AND C.Z<15;

    (d) SELECT A.* FROM A,B,C,D WHERE A.W<5 AND A.X=B.X AND A.X=C.X AND C.Y=D.Y AND D.Z<>7;

    (e) SELECT A.X,B.Y FROM A,B,C WHERE A.X=B.X+C.X AND B.Y<C.Z;

    (f) SELECT A.X,A.Y FROM A,B,C WHERE A.X=B.X OR A.Y=C.Y;

3. ¿En qué consiste el problema de programación dinámica que enfrenta el optimizador?

4. Considere un evaluador de consultas que realice *en el vuelo* las operaciones posteriores a un join (hasta que aparece otro join) y que se realiza Bucle Anidado Simple (iterar sólo por tuplas). Sean A(X,Y), B(X,Z), C(Y,Z) tres tablas con 3000, 2000 y 1000 registros respectivamente. X, Y, Z son atributos numéricos entre 0 y 99 (100 valores distintos). Suponiendo que los valores de los atributos están distribuidos uniformemente por los registros (por ej. $E(C.X = k) = 10, \forall k \in \{0, ..., 99\}$), diga cuál es la mejora relativa en las siguientes consultas respecto a la versión sin optimizar:

(a) SELECT A.X,A.Z FROM A,B,C WHERE A.X=B.X AND B.Y=C.Y AND C.Z<25;

(b) SELECT A.* FROM A,B,C WHERE B.X=C.X AND A.Y=C.Y AND (A.Z=5 OR B.Z=5);

(c) SELECT A.X,A.Z FROM A,B,C WHERE A.X=B.X+C.X AND C.Z=5 AND B.Z=10 AND B.Y>C.Y;

5. Agregue la siguiente regla heurística a las vistas en clase: "dar prioridad a los equijoin". ¿Cómo mejora cada una de las consultas anteriores? Dé números.

2.2 Respuestas selectas

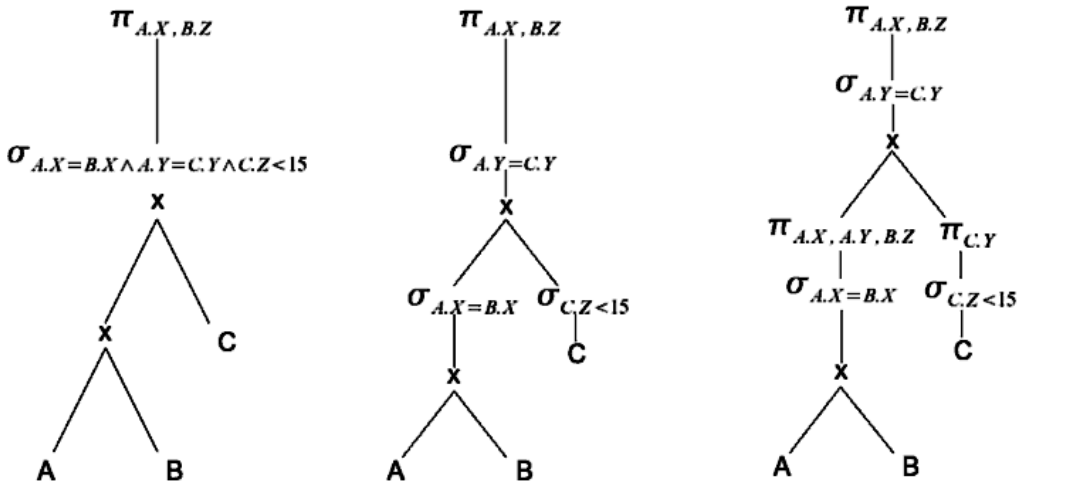
1. —

2. Las reglas heurísticas vistas en clase consisten en: expresar la consulta en forma canónica, separar las σ (selecciones) y llevarlos lo más cerca posible de las hojas, separar las π (proyecciones) y dejarlos justo después de los σ (para que se evalúen después). En la evaluación, las σ (selecciones) cercanos a los $\times$ (productos cartesiano) se convertirán en $\bowtie$ (joins) y la inmediata aplicación de π (proyecciones) reduce el uso de memoria secundaria al escribir una respuesta más pequeña (si aplica).

   (a) —

   (b) —

   (c) Ejercicio resuelto. Úselo como guía para el resto de los ejercicios pues, en efecto, la estrategia es la misma para todos los ejercicios. Nota: una variante de la forma final consiste en aplicar proyección también sobre las tablas sobre las que no hay otro proceso (A y B tendrían proyecciones).



   (d) —

   (e) —

   (f) —

3. En minimizar el costo de la evaluación de la consulta a la vez que la diseña.

4. Bucle anidado simple implica que habrá tantos I/O como registros.

   (a) Árbol canónico: $A \times B$ cuesta leer A y B por registros (3000 + 2000 = 5000) y escribir A*B (3000 * 2000 = 6000000). $(A \times B) \bowtie C$ cuesta leer ambas partes PERO haciendo join; bucle anidado simple implica leer cada registro de $A \times B$ y compararlo con cada uno de C (o viceversa), así que la lectura cuesta toda la comparación (6000000*1000 = 6000000000). La escritura cuesta tanto como el total de la respuesta, A.X=B.X y B.Y=C.Y y C.Z<25. Dado A.X, hay 20 B.X iguales. Dado B.Y, hay 10 C.Y iguales. Un cuarto de los C.Z cumplen con C.Z<25. El costo de escribir es 150000. Costo original: 5000 + 6000000 + 6000000000 + 150000 = 6006155000.
   Forma optimizada (por heurística): $A \bowtie B$ cuesta leer A y B comparando (6000000) pero sólo se escribe lo que cumple A.X=B.X (3000*20 = 60000). C.Z<25 implica leer C y guardar sólo 1/4 de C (1000+250=1250). $(A \bowtie B) \bowtie C$ cuesta leer ambas respuestas anteriores comparando (60000*250 = 15000000) y escribir el resultado final (60000 * 250/100 = 150000, igual que el caso no optimizado). Costo optimizado: 6000000 + 60000 + 1250 + 15000000 + 150000 = 21211250. Comparación: la evaluación original demora 283 veces la evaluación optimizada heurísticamente.

   (b) —

   (c) —

5. Esta regla significa que debemos hacer lo posible para que los equijoin estén en la parte inferior izquierda de los árboles de evaluación de consulta.

   (a) No cambia.

   (b) Cambia. Propuesto el cálculo.

   (c) —

3 Evaluación de operadores relacionales

3.1 Problemas

1. ¿Cuáles son las principales estrategias para evaluar operadores relacionales?

2. ¿En qué consisten los siguientes bucles anidados para el join?

   (a) Bucle anidado simple.

   (b) Bucle anidado por bloques.

   (c) Bucle anidado por índices.

3. ¿En qué consisten sort-merge join (SMJ) y hash join (HJ)?

4. Resuelva el problema 4 de la sección anterior considerando un tamaño de 1 Kb por registro y un tamaño de 32 Kb por página, y el join realizado por Bucle Anidado por Bloques. ¿Cómo cambiaría todo ante un Bucle Anidado por Índices si los atributos X de A, B y C estuvieran indexados con B+tree?

3.2 Soluciones a problemas selectos

1. Son *indexación*, *iteración* y *partición*. Indexación consiste en usar un índice existente en la evaluación. Iteración consiste en repasar todas las tuplas de la relación, aunque si un índice contiene los atributos que se desean evaluar, entonces se revisa todo el índice. Partición consiste en trabajar las tuplas para hacer más eficiente una operación mediante ordenamiento o aplicación de hashing.

   (a) En general sólo se les llama nested loop join, pero hay una gran variedad de éstos. En particular:

(b) Como un doble for sobre los registros de cada tabla.

(c) Como el anterior, sólo que se recorre por bloques de disco.

2. Como su nombre lo dice, una tabla se recorre por registros o bloques (conviene la segunda opción) y la otra se recorre usando el índice existente (ésta es la manera tradicional de usar los índices en un join).

3. —

4. —