Bases de datos: Ayudantía I2

Romano Fenzo - rfenzo@uc.cl

12/10/2018

1 Repaso

1.1 Clustered Index

- Define el orden de los datos
- Solo se puede tener un indice clustered
- Entrega la tupla buscada
- Ejemplo: Libreta de teléfonos

1.2 Unclustered Index

- Crea tabla complementaria con puntero a los datos reales
- Se pueden tener multiples indices unclustered
- Entrega un puntero a la tupla buscada, y por tanto requerirá un acceso a memoria adicional para obtener la tupla.
- Ejemplo: Índice por palabras al final de un libro

1.3 Hash Index

- Se deben manejar colisiones, por ejemplo con listas ligadas
- Bueno para *queries* de igualdad, malo para las de rangos

$$\textbf{Costo de I/O} \sim \mathcal{O}\left(\frac{|\text{Records}|}{R \cdot P}\right)$$

$$\text{R} = \frac{\text{Records}}{\text{Páginas}} \quad \text{P} = \frac{\text{Páginas}}{\text{Buckets}}$$

Nota: Por lo general se asume que el costo es igual a uno

1.4 B+ Tree Index

- Mantiene balanceado el arbol para garantizar profundidad logarítmica
- Bueno para *queries* de igualdad (no tanto como hash index) y rangos

$$\textbf{Costo de I/O} \sim \mathcal{O}\left(\log_{R/2}\left(\frac{2 \cdot |\text{Records}|}{R}\right)\right)$$

$$\text{R} = \frac{\text{Records}}{\text{Páginas}}$$

1.5 Nested Loop Join

Consiste en iterar sobre R y luego sobre S, teninedo que recorrer S tantas veces como tuplas tenga R

$$\text{R} \bowtie \text{S} \rightarrow \text{Costo I/O} = \text{Costo(R)} + \text{Tuplas(R)} \cdot \text{Costo(S)}$$

1.5.1 Block Nested Loop Join

Si S tiene muchas tuplas, lo anterior es ineficiente. En mejor cargar un buffer con páginas de R e iterar S una sola vez para comparar con ese conjunto de tuplas. Luego volver a llenar el buffer y seguir hasta que no que no queden tuplas en R.

$$\text{R} \bowtie \text{S} \rightarrow \text{Costo I/O} = \text{Costo(R)} + (\text{Páginas(R)}/|\text{Buffer}|) \cdot \text{Costo(S)}$$

1.6 EXTRA (para I3): External Merge Sort

- Gran cantidad de datos → No caben todos en la RAM
- Cada fase requiere I/O de las N páginas a disco
- *run*: Colección de páginas ordenadas
- Fase 0: Cada pagína es ordenada con algún algoritmo (Ej. *quicksort*)
- Fases ≥ 1: Hacen merge de K *runs*

$$\textbf{Costo de I/O} = 2 \cdot N \cdot \#\text{Fases}$$

1.6.1 Algoritmo base:

Buffer de 2 + 1 pagínas y runs iniciales de 1 página:

$$\#\ \textbf{Fases} = 1 + \lceil \log_2(N) \rceil$$

1.6.2 Algoritmo optimizado en I/O:

Buffer de B + 1 páginas y runs iniciales de B + 1 páginas:

$$\#\ \textbf{Fases} = 1 + \left\lceil \log_B \left\lceil \frac{N}{B+1} \right\rceil \right\rceil$$

Buffer óptimo:

$$\#\ \text{Fases} = 2 \rightarrow B \geq \sqrt{N}$$

Costo de I/O con buffer óptimo:

$$\text{Costo} = 2 \cdot N \cdot \#\text{Fases} = 4 \cdot N$$

Y si el último resultado no lo escribimos en disco?

$$\text{Costo} = 4 \cdot N - N = 3 \cdot N$$

2 Ejercicios

2.1 Guía I2 - Pregunta 4: B+ Tree, Join indexado, psycopg2

b) Sea la relación R(a, b, c, d) cuyo tamaño es de 2 millón de tuplas, en que cada página contiene en promedio P tuplas. Las tuplas de R están ordenados de manera aleatoria (cuando no hay índice). El atributo a es además un candidato a llave priamria, cuyos valores van del 0 al 1.999.999 (distribuidos uniformemente). Para cada una de las consultas a continuación, diga el número de I/O que se harán en cada uno de los siguientes casos:

- Analizar R sin ningún índice
- Usar un B+Tree Unclustered sobre el atributo a. El árbol es de altura h. La cantidad de punteros por pagina es de M . El índice de ocupación de las hojas es del 90%

Las consultas son:

1. Encontrar todas las tuplas de R.

2. Encontrar todas las tuplas de R tal que a < 100.

3. Encontrar todas las tuplas de R tal que a $= 100303$.

4. Encontrar todas las tuplas de R tal que a > 50 y a ≤ 150.

Comente sus resultados e indique que índice prefiere en los distintos casos.

c) En clases se vio que el costo en I/O del Block Nested Loop Join es de Pags(R) + (Pags(R)/Buffer) · Pags(S). Existen otros algoritmos de join que pueden hacer usos de índices sobre las bases de datos. Entregue un algoritmo (explicado con palabras) que compute R $\bowtie$ S, con R(a, b) y S(a, c) en el que ambas llaves primarias están indexadas por un B+Tree clustered. Entregue el costo estimado en I/O (debería ser mejor que el costo del Block Nested Loop Join).

Hint: El B+Tree mantiene la relación ordenada. Debe hacer uso de esto.

d) Considere las relaciones R(a int) y S(a int), en donde las relaciones están indexadas por un B+Tree clustered. Tienes acceso a Python y psycopg2, pero sólo puedes hacer consultas a la base de datos del tipo:

- CREATE TABLE
- SELECT *
- INSERT INTO

Entregue un algoritmo en Python que haga uso la librería **psycopg2** que calcule la consulta SELECT * FROM R EXCEPT SELECT * FROM S, pero que lea cada tupla una única vez. Asuma que los valores en las relaciones son entregados según el orden en que están almacenadas por los B+Tree. Además asuma que los valores no caben en memoria.

2.2 Guía I3 2017-2 - Pregunta 1.e: Hash Index

Considere los datos de la pregunta 4.b de la Guía I2. Suponga que P es cercano a 5. Considere un Hash Index Clustered con 200.000 Buckets. ¿Cuanto I/O requiere la consulta "Encontrar todas las tupals de R tal que $a = 100303$"?. ¿Cómo puede ayudar un Hash Index dinámico?.

2.3 Guía PL/pgSQL - Pregunta 1: Secuencia de números con intervalo

Hacer un procedimiento almacenado que reciba min, max y delta, los cuales séan usados para entregar una tabla con los números entre min y max separados por el intervalo delta.
Ej. seq(1,8,2) retorna una tabla con los valores 1,3,5,7.

2.4 I2 2017-2 - Pregunta 3: Recursión

Suponga que usted tiene una tabla Caminos(ciudad_origen varchar(100), costo int, ciudad_destino varchar(100)). Esta tabla representa a las ciudades que están conectadas por un camino desde *ciudad_origen* hasta *ciudad_destino* con un costo asociado (por peajes, combustible, entre otros) esto equivale al valor del *costo*.

Entregue una consulta SQL que para un número C dado, entregue todas las ciudades alcanzables entre si cuyo costo acumulado sea menor o igual que C. Por simplicidad asuma que el grafo generado por las ciudades es dirigido, acíclico y que no hay dos formas de llegar a una ciudad.