Pauta

IIC2413 – Bases de Datos

Control 2 - Indices y algoritmos

El plazo para la entrega es el viernes 14 de octubre, hasta las 23:59. La entrega debe hacerse en un **pdf** y subirse a Canvas (se creará una tarea con el form para subir la entrega).

Pregunta 1 (2 ptos)

Considere la relación

`Cervezas(id int PRIMARY KEY, nombre varchar(20), marca varchar(20), estilo varchar(15), precio int)`

indexada con un hash-join clustered sobre el atributo `id`, un hash-join unclustered sobre marca y un B+tree unclustered sobre precio.

1. Indique el costo I/O de las siguientes consultas asumiendo que la relación tiene 10000 tuplas, que el id va desde el 0 hasta el 9999, que en una pagina caben 80 tuplas o 250 punteros, que la altura del B+Tree es h, que sus hojas están ocupadas a un 60 % y que el hash index tiene solo una pagina por bucket.

   - `SELECT * FROM Cervezas WHERE id = 20`
   - `SELECT * FROM Cervezas WHERE id <=3`
   - `SELECT * FROM Cervezas WHERE precio >2000 AND precio <3000` (asuma que existen menos de 80 tuplas que cumplen esta condición)
   - `SELECT * FROM Cervezas WHERE marca = 'Tamango Brebajes'`
   - `SELECT * FROM Cervezas WHERE estilo = 'Hazy IPA'`

   } 0.3 c/u

2. Bajo las mismas condiciones de la pregunta anterior, explique como podría resolver la consulta `SELECT * FROM Cervezas WHERE precio <3000 AND marca = 'Tamango Brebajes'` (haciendo un uso inteligente de los indices) e indique el costo I/O de su propuesta. Asuma que la cantidad de cervezas que valen menos de 3000 caben en una página. → 0.5.

Pregunta 2 (2 ptos)

Considere que tiene las relaciones `R(a int PRIMARY KEY, b int)` y `S(b int PRIMARY KEY, c int)`. Escriba en pseudo-código un programa para computar $R \bowtie S$ tal que recorra R y S solo una vez. *Hint: es probable que tenga que pre-procesar la(s) relacion(es) de alguna manera...*

Pregunta 3 (2 pts)

Considere las relaciones `R (a int, b int)` y `S(a int, c int)` e imagine que casi todas las consultas que se hacen implican hacer un join entre R y S. Imagine también que R ya se encuentra indexada según el atributo b. Responda lo siguiente:

1. Explique cómo resolver $R \bowtie S$ y cuál es su costo en operaciones I/O 0.5
2. Recomendaría usted incluir algún otro índice en las relaciones? Si la respuesta es sí, indique el tipo de índice y cómo quedaría el costo I/O de la consulta. Si la respuesta es no, indique por qué no. 1.0.
3. ¿Existe alguna manera en la que se puede disminuir el costo I/O de la consulta sin crear nuevos índices? Justifique su respuesta. 0.5

Pregunta #1

1. Indique el costo I/O de las siguientes consultas asumiendo que la relación tiene 10000 tuplas, que el id va desde el 0 hasta el 9999, que en una pagina caben 80 tuplas o 250 punteros, que la altura del B+Tree es h, que sus hojas están ocupadas a un 60 % y que el hash index tiene solo una pagina por bucket. (1.5 pts)
   - `SELECT * FROM Cervezas WHERE id = 20`
   - `SELECT * FROM Cervezas WHERE id <=3`
   - `SELECT * FROM Cervezas WHERE precio >2000 AND precio <3000` (asuma que existen menos de 80 tuplas que cumplen esta condición)
   - `SELECT * FROM Cervezas WHERE marca = 'Tamango Brebajes'`
   - `SELECT * FROM Cervezas WHERE estilo = 'Hazy IPA'`

(0.3 pts cada una)

- SELECT * FROM cervezas WHERE id = 20

  El costo es O(1) pues usamos el hash-index

- SELECT * FROM cervezas WHERE id <= 3

  El costo es 4, pues en el peor caso cada tupla estará en un bucket distinto.

- —. precio > 2000 AND precio < 3000

  Aquí usamos el B+tree, el costo es

$$\underbrace{h}_{\text{altura del árbol}} + \underbrace{1}_{\text{página donde están los punteros}} + \underbrace{X}_{\text{tuplas que cumplen la condición (son menos de 80)}}$$

↳ podríamos acotarlo por $h + 81$ o $h + 80$

- ... marca = 'Tamango Brebajes"

↓

marca está indexada en un hash-index clustered, entonces el costo I/O es

$1 + X$

X = cantidad de tuplas que cumplen la condición

- ... estilo = "Hazy IpA"

↓

como no hay índice sobre estilo, hay que recorrer todas las páginas

$\Rightarrow$ el costo es $\frac{10.000}{80} = 125$

2. Bajo las mismas condiciones de la pregunta anterior, explique como podría resolver la consulta `SELECT * FROM Cervezas WHERE precio <3000 AND marca = 'Tamango Brebajes'` (haciendo un uso inteligente de los indices) e indique el costo I/O de su propuesta. Asuma que la cantidad de cervezas que valen menos de 3000 caben en una página. (0.5 pts)

lo esto es lo mismo que la penultima consulta, busco la marca en el hash-index clustered y luego sigo cada puntero ...
la única duf. es que tengo que revisar una condición más, pero eso no afecta el costo I/O

Costo I/O $1 + X$

* Otra opción de la pregunta 1.2 sería usar el BNL en vez, en cuyo caso el costo sería $h + X$

b # de averías en vez < 3000.

Pregunta 2

Considere que tiene las relaciones `R(a int PRIMARY KEY, b int)` y `S(b int PRIMARY KEY, c int)`. Escriba en pseudo-código un programa para computar $R \bowtie S$ tal que recorra R y S solo una vez. *Hint: es probable que tenga que pre-procesar la(s) relacion(es) de alguna manera...* 2 pts.

La idea es simple: primero hay que ordenar las relaciones y después hacer lo siguiente

```
R.open()
S.open()
x = R.next()
y = S.next()
while x ≠ NULL AND y ≠ NULL:
    IF π_b(x) = π_b(y):
        OUTPUT x
        x = R.next()
        y = S.next()
    ELIF x < y:
        x = R.next()
    ELIF y < x:
        y = S.next().
```

como están ordenados, solo avanzo en el mismo cuando este se llama sort merge join

Pregunta 3

Considere las relaciones `R (a int, b int)` y `S(a int, c int)` e imagine que casi todas las consultas que se hacen implican hacer un join entre R y S. Imagine también que R ya se encuentra indexada según el atributo b. Responda lo siguiente:

1. Explique cómo resolver $R \bowtie S$ y cuál es su costo en operaciones I/O 0.5 pts
2. Recomendaría usted incluir algún otro índice en las relaciones? Si la respuesta es sí, indique el tipo de índice y cómo quedaría el costo I/O de la consulta. Si la respuesta es no, indique por qué no. 1 pt
3. ¿Existe alguna manera en la que se puede disminuir el costo I/O de la consulta sin crear nuevos índices? Justifique su respuesta. 0.5 pts .

3.1) Podemos hacer un nested loop join cuyo costo es

$$\text{Costo}(R) + \text{tuplas}(R) \cdot \text{Costo}(S)$$

→ acá los estudiantes pueden proponer hacer un block nested loop join.

3.2) Lo más lógico sería indexar R, S o ambos según el atributo ~~b~~ a. En el caso de R el index sería un clustered, en el caso de S, como no hay otro index, podría ser clustered.

Si usamos hash index en ambos, el costo podría ser

$$\text{Costo}(R) + \text{tuplas}(R)$$

ó

$$\text{Costo}(S) + 2 \cdot \text{tuplas}(S)$$

} si usan un B+ tree podría ser distinto...

3.3) Una manera de agilizar la consulta sin nuevos índices sería normalizar las relaciones...