Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Primer Semestre de 2019

IIC 2413 – Bases de Datos
Interrogación 3

Pregunta 1: Índices, algoritmos y transacciones

a) [2 ptos] El B+Tree explicado en clases no está preparado para trabajar con duplicados. Una posible solución que al llegar a las hojas nos encontremos con un puntero a una lista de punteros (que se almacenan en una página de disco) en vez de un puntero a la tupla. Si un valor llega a tener muchos duplicados y los punteros no caben en una página tenemos que utilizar overflow pages. Para las consultas de igualdad aparte de bajar por el árbol debemos considerar recorrer todas las overflow pages del valor. Para las consultas de rango debemos considerar las overflow pages de todos los valores en el rango. Dado que el índice es *unclustered* debemos seguir sumando la cardinalidad del output de la consulta.
b) [2 ptos] La relación tiene 1 millón de tuplas, y en cada página caben 50 tuplas y 200 punteros. El árbol es de altura h y las hojas están ocupadas al 75 %. El Hash Index tiene una página por *bucket* (i.e. es suficientemente grande y no tiene *overflow pages*), y cada edad va a parar a un *bucket* distinto.

Los costos son los siguientes:

1. `SELECT * FROM Persona WHERE id=4444`: $h + 2$.[1]
2. `SELECT * FROM Persona WHERE nombre='Claudio Bravo'`: 20000, porque se recorre la relación entera.
3. `SELECT * FROM Persona WHERE edad=33`: 1.
4. `SELECT * FROM Persona WHERE id<10000`: $h + \lceil \frac{10000}{200} \rceil + 10000$
5. `SELECT * FROM Persona WHERE edad>=25 and edad<35`: 10.

c) [2 ptos] Ordenar previamente las relaciones nos sirve porque al realizar el join solamente recorremos cada relación una vez. No sería necesario ejecutar un algoritmo de ordenamiento si las relaciones hubiesen estado indexadas por el atributo `b` con un B+Tree. Una opción para el costo es sumar el costo de ordenar y hacer el join. Esto sería $4|R| + 4|S| + |R| + |S| = 5(|R| + |S|)$. Sin embargo, dado que lo que queremos es hacer el join, podemos sencillamente completar solamente la fase 0 del ordenamiento y pasar directamente a hacer el join con los resultados parciales en vez de terminar con el ordenamiento. El costo real de hacer esto sería $3(|R| + |S|)$.

Bonus [2 décimas en la interrogación] En clases un alumno expuso una teoría de por qué Arturo Prat había sido elevado a héroe nacional a pesar de haber perdido el Combate Naval de Iquique. Explique esta teoría brevemente.

[1] La respuesta también puede ser $h + 1$ en el caso de considerar que al descender h nos encontramos con el puntero.

Pregunta 2: Transacciones y Logs

Transacciones

a) [1 pt] Cuando X es `READ` es claro que no se forma un ciclo por lo tanto el *schedule* es *conflict serializable*. Si X es `WRITE` se forma un ciclo y el *schedule* no es *conflict serializable*. Es necesario mostrar el grafo en esta pregunta.

b) [2 pts] Sabemos que todas las acciones conflictivas involucran una operación de escritura. Cuando usamos Strict-2PL, cada vez que una transacción $T1$ desea realizar una operación de escritura sobre un atributo, digamos A, en el caso de que otra transacción $T2$ quiera leer o escribir A tenemos dos opciones:

- Si $T2$ hizo esto antes de $T1$, necesariamente vamos a tener que esperar a que $T2$ termine para que libere todos sus locks, por lo que la arista en el grafo iría solamente desde $T2$ a $T1$ y se hace imposible que haya una arista en el otro sentido en el futuro.
- Si $T2$ hizo esto depués de $T1$, sabemos que $T1$ ya terminó porque para poder ejecutar la acción conflictiva de $T2$ necesitamos que $T1$ libere el XLock, por lo que solamente puede haber una arista en el grafo desde $T1$ a $T2$ y no al revés.

Podemos extender este mismo argumentos para ciclos de tamaño más grande. La idea es que una transacción que aparece después intentando hacer algo conflictivo no puede tener una arista a una transacción que apareció antes porque esta ya tuvo que haber terminado para poder liberar sus *locks*.

Redo Logging

Las respuestas son las siguientes:

a) Desde qué parte del *log* debo comenzar el proceso de *redo*: hasta el `<START T2>`.

b) Qué variables deben rehacer sus cambios y cuál es el valor con el que quedarán: solamente `T3`. Rehace el valor de `a` a 15 y de `d` a 23.

c) Qué variables (de las que aparecen en el *log*) no son cambiadas en el proceso: `b`, `c` y `e`.

d) Si no hubiesemos encontrado la línea `<END CKPT>`, ¿desde qué parte del *log* debería comenzar el proceso de *redo*?: hay que leer el log entero. Si no hay un *end checkpoint* no tenemos seguridad de que `T1` esté efectivamente persistente en disco.

Pregunta 3: MongoDB

En esta pregunta hay que asegurarse de crear un índice por texto al contenido de los comentarios y a la descripción del fotograma.

- **a)** Entregue el *id* de fotograma, junto al link del fotograma, junto al número de comentarios que tiene para cada fotograma que contiene el texto "#Cobreloa" y que no tenga el texto "Cobreloa perdió" en su descripción de fotograma:

  Hacemos una búsqueda por texto sobre la descripción del fotograma según lo solicitado. Para cada uno de estos comentarios obtenemos el `id` del fotograma y hacemos una consulta a la colección de fotogramas para obtener su `url`.

- **b)** Entregue el *id* del fotograma, junto al número total de "Me Gusta" de todas las fotos que hayan sido comentadas por el usuario con *id* 23:

  Podemos hacer la consulta en la colección de comentarios cuando el `id` de la persona que comenta es el 23. Luego para cada uno de aquellos `id` de fotograma hacemos la consulta sobre la colección de usuarios en que el arreglo de `me_gusta` contenga aquel `id` de fotograma y sumamos todos aquellos usuarios.

- **[2 pts]** Entregue todos los documentos de la colección Fotograma (se debe entregar el documento con todos sus atributos) que no tengan comentarios con la frase "Vamos Chile Bicampeón":

  Hacemos la búsqueda por texto en los comentarios según lo solicitado para obtener cada uno de los `id` de fotograma. Luego hacemos una consulta por cada `id` de fotograma obtenido desde los comentarios para traer cada uno de los documentos de la colección fotograma.