Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Segundo Semestre de 2017

IIC 2413 – Bases de Datos
Interrogación 3

Pregunta 1: Almacenamiento, índices y algoritmos

[0.5 pts] ¿Qué ventaja tiene una base de datos columnar sobre una estándar para las consultas de agregación?

[1 pt] En clases vimos que el costo de ordenar una tabla de N páginas era $4N$. Explique de donde proviene este costo y en base a eso indique una forma de reducir el costo a $3N$.

[2.5 pts] Suponga que tiene una relación $R(a, b)$ que tiene 1 millón de tuplas originalmente ordenadas aleatoriamente. El atributo a es candidato a llave primaria con valores que van del 0 al 4.000.000 distribuidos uniformemente. Además, en una página caben P tuplas. Suponga las siguientes consultas:

- `SELECT * FROM R`
- `SELECT * FROM R WHERE a=1000345`
- `SELECT * FROM R WHERE a >=0 AND a <100000`

Indique el costo I/O para cada una de las consultas cuando el atributo a está indexado con:

- Un B+Tree Clustered
- Un B+Tree Unclustered
- Un Hash Index Clustered
- Un Hash Index Unclustered

Para los B+Tree asuma que las hojas están ocupadas al 60 %. Para los Hash Index asuma que dispone de B buckets y que la función de Hash distribuye los datos uniformemente en los buckets. Además en el caso de los índices Unclustered, los punteros que caben en una página son M.

[2 pts] Suponga una relación $R(a, b)$ de N páginas en donde el atributo a es candidato a llave primaria. Suponga que indexa la relación por ese atributo con un Hash Index Clustered con B buckets. Conteste:

- Por qué en el peor caso (piense en una mala función de *hash*) el costo en I/O de la consulta $\sigma_{a=i}(R)$ puede llegar a ser N.
- Por qué en el mejor caso el costo en I/O es 1.
- Por qué en con una función de *hash* uniforme, usted esperaría que el costo fuera N/B.
- Si el Hash Index es dinámico, ¿cuál esperaría que fuese el costo de la consulta?

[0.6 pts] Bonus: Deporte UC - Finales LDES Talca 2017
¿Qué equipo de vóleibol femenino ganará el nacional que se está desarrollando en Talca?

1. PUC
2. UChile
3. UNAB
4. U. de Conce

Pregunta 2: Algoritmos, transacciones y Map Reduce

[2 pts] ¿Por qué el tiempo de respuesta de un Block Nested Loop Join puede llegar a ser mucho menor que el de un Nested Loop Join? Explique en términos de uso de buffer y costos de I/O. Además indique como un Join del tipo $R(a,b) \bowtie S(b,c)$ puede verse beneficiado si ambas relaciones tienen indexado el atributo b con un B+Tree.

[3 pts] Considere el Schedule del cuadro 1. Diga si es o no *conflict serializable*. En caso de que no lo sea, explique por qué e indique cómo Strict-2PL puede resolver el problema.

T1	T2	T3	T4
R(a)			
	R(b)		
		W(a)	
			R(a)
	R(a)		
	R(d)		
W(d)			

Cuadro 1: schedule problema 2.

[1 pt] Considere un esquema de dos tablas `A(id int, name varchar(10))` y `B(id int, name varchar(10))`. Suponga que quiere hacer la consulta en álgebra relacional $\pi_{A.name,B.name}(A \bowtie B)$ pero solamente dispone de un archivo que se ve de la siguiente forma:

```
A,1,palabra1
A,2,palabra2
A,3,palabra3
B,1,palabra1
...
```

El primer término antes de la coma representa la tabla, el segundo el id y el tercero el name. Explique con palabras un algoritmo Map - Reduce que ejecute la consulta deseada.

Pregunta 3: MongoDB

Piense en una base de datos en MongoDB con dos colecciones, una de usuarios de una red social y otra de estados publicados en ella:

```
// Usuarios
{
  "uid": 1,
```

```
  "name": "Fernando Pieressa",
  "age": 22,
  "description": {
    "estudia_en": "PUC",
    "Películas favoritas": ["Interstellar", "Blade Runner"]
  }
}

// Estados
{
  "eid": 1,
  "uid": 1,
  "content": "Gigante Cobreloa #VamosZorros"
  "likes": [1, 4, 7]
}
```

En que los usuarios tienen anidado un documento `description` que indica dónde estudian y sus películas favoritas (con un JSON Array que contiene los labels de las películas). Además cada estado emitido por el usuario posee un arreglo con los `id` de los usuarios que le han dado like a ese estado.

Se pide que entregue la siguiente consulta en MongoDB (sin usar un lenguaje de programación externo):

- **[1 pt]** Entregue el `id` y nombre de todos los usuarios que estudian en la "PUC" y tienen más de 20 años.

Ahora utilizando PyMongo o JavaScript se pide un procedimiento que entregue lo siquiente:

- **[2 pts]** Entregue cada a usuario junto al número total de likes que tiene[1].

Ahora se pide que entregue una secuencia de pasos para crear el índice correspondiente, junto al procedimiento para responder la siguiente consulta:

- **[3 pts]** Imprima el id de todos los estados que contienen el hashtag "#VamosZorros" pero no el hashtag "#VamosColoColo", junto al nombre de todos los usuarios que le dieron like al estado.

[1] En JavaScript puede obtener el largo de un arreglo `a` con la instrucción `a.length`