Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Segundo Semestre de 2016

IIC 2413 – Bases de Datos
Interrogación 3

Pregunta 1: Almacenamiento, índices y Map Reduce

a) [0.5 ptos] Explique qué es un sistema de bases de datos orientada a columnas. Indique bajo qué circunstancias me conviene utilizar una sistema de este tipo.

b) [0.5 ptos] Explique cómo el tamaño del track puede influir en el tamaño de un bloque de disco.

c) [1.5 ptos] Considere una relación de 3.000.000 páginas. Indique el número de fases y de I/O del External Merge Sort optimizado visto en clases cuando:

- En Buffer caben 11 páginas.
- En Buffer caben 101 páginas.
- En Buffer caben 1001 páginas.

Comente sus resultados. Además diga a cuanto converge el número de I/O cuando el buffer sigue creciendo. Indique que pasa si se evita escribir el resultado de la última fase y el resultado se imprime directamente en consola.

d) [2.5 ptos] 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 primaria, cuyos valores van del 0 al 1.999.999 (distribuídos 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 70 %.
- Usar un *Hash Index Clustered* con 1 millón de buckets.

Las consultas son:

1. Encontrar todas las tuplas de R.
2. Encontrar todas las tuplas de R tal que $a < 70$.
3. Encontrar todas las tuplas de R tal que $a = 207890$.
4. Encontrar todas las tuplas de R tal que $a > 50$ y $a \leq 200$.

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

e) [1 pto] Considere un esquema de dos tablas `A(id int, name varchar(10))` y `B(id int, name varchar(10))`. Suponga que quiere hacer el *Left Outer Join* entre A y B ($A \rtimes B$) comparando según id, 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. Entregue un algoritmo (basta explicar con palabras) Map - Reduce que ejecute la consulta deseada. Para indicar un valor nulo puede entregar el string "_".

Pregunta 2: Algoritmos internos y transacciones

a) [3 ptos] En clases se vio que el costo en I/O del Block Nested Loop Join es de $Pags(R)+(Pags(R)/Buffer)\cdot Pags(S)$. Existen otros algoritmos de join que pueden hacer usos de índices sobre las bases de datos.

- Entregue un algoritmo (explicado con palabras) que compute el join $R \bowtie S$ entre `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.
- ¿Cómo cambia y de qué depende el costo cuando el B+Tree es Unclustered? ¿Bajo qué instancias me conviene usar el Block Nested Loop Join vs Hacer el join con el B+Tree Unclustered? No es necesario que indique el costo del B+Tree explícitamente, puede explicar lo que sucede con palabras.
  Hint: No todas las tuplas de R y S forman parte del output. El principal costo de un índice unclustered es tener que ir a buscar lo que indican los punteros a disco.

b) [3 ptos] Sea el schedule del cuadro 1:

- **[1 pto]** Entregue el grafo de precedencia.
- **[0.5 ptos]** Determine si es *Serializable*. Fundamente su respuesta e indique qué significa que sea searializable.
- **[1.5 ptos]** ¿Por qué al utilizar *Strict 2PL* los schedules necesariamente son *Serializables*? Explique qué pasa en este schedule en concreto sin entregar otro grafo (puede explicar con palabras).

c) Bonus [0.3 ptos] Indique:

- Quién fue Jim Gray.
- Cuál fue su aporte a las bases de datos.
- Qué le pasó.

T1	T2	T3	T4	T5
		R(c)		
	R(d)			
				W(c)
			W(a)	
W(b)				
	W(f)			
				R(d)
	W(b)			
R(c)				
		R(a)		
R(d)				
			R(f)	
			W(f)	

Cuadro 1: Schedule pregunta 2 parte b

Pregunta 3: MongoDB

Piense en una base de datos en MongoDB con dos colecciones, una de personas, otra de compras y otra de posts en una red social. Se muestra a continuación una instancia de documento para cada colección:

```
// Persona
{
  "id_persona": 1,
  "name": "Florencia Barrios"
}

// Post
{
  "id_post": 1,
  "id_persona": 1,
  "content": "Mañana juego un torneo de hockey!"
}

// Compra
{
  "id_compra": 1,
  "id_persona": 1,
  "fecha": "10-09-2016",
  "compras": [
    {
      "producto": "Palo de Hockey",
      "tipo": "Deporte",
      "valor": 20000
    },
    {
      "producto": "Café con leche",
      "tipo": "Comida",
      "valor": 1000
    }
```

```
  ]
}
```

Es importante notar que una persona puede tener asociado más de un documento de compra y post (por ejemplo, el documento con `id_compra` 2 puede pertenecer también a la persona con id 1). Entregue las siguientes consultas en MongoDB. Si va a utilizar un índice indique cómo lo creó. Puede hacer uso de JavaScript o Python en caso de ser necesario:

- **[1.5 ptos]** Entregue los posts que contenga la palabra "Gol" y la frase "Vamos Chile" pero no la frase "Vamos Colombia" junto al nombre del usuario que lo emitió.
- **[1.5 ptos]** Entregue cada persona junto al nombre de cada producto tipo "Deporte" que ha comprado.
- **[3 ptos]** Para cada persona que haya emitido un post que contenga la frase "vamo a Calmarno" y no la frase "vamo a evolucionarlo" indique el nombre de la persona junto al total gastado comprando productos de tipo "Pokémon".