Base de Datos (75.15/95.05/TA044)

Segundo Parcial Promocional

TEMA	**Proc. de Cons.**			**Fecha:** 26 de junio de 2024
2024141	**NoSQL**			**Padrón:** _____
	Conc. y Rec.			**Apellido:** _____
Corrigió:				**Nombre:** _____
Nota:				**Cantidad de hojas:** _____
☐ Aprobado ☐ Insuficiente				

Criterio de aprobación: El examen está compuesto por 6 ítems, cada uno de los cuales se corrige como B/B-/Reg/Reg-/M. El examen se aprueba con nota mayor o igual a 4(cuatro) y la condición de aprobación es desarrollar un ítem bien (B/B-) en los 3 grupos de ejercicios (procesamiento de consultas, NoSQL, concurrencia/recuperación). Adicionalmente, no deberá haber más de dos ítems mal o no desarrollados.

1. *(Procesamiento de consultas)* La empresa multinacional *BueyStar* quiere obtener los clientes de dos países, utilizando la siguiente consulta:

   - `Clientes(id_cliente, nombre, email, pais, fecha_creacion, ...)`

   ```
   SELECT * FROM Clientes WHERE pais = 'Argentina' OR pais = 'Uzbekistan';
   ```

   En la tabla existe un índice de clustering por la columna pais con una altura de 2, y además se cuenta con un histograma de los 3 valores más frecuentes y su cantidad de filas.

   Se pide:

   a) Calcule, indicando el costo en acceso a bloques de disco para cada estrategia, si conviene utilizar el índice para resolver la consulta o si conviene efectuar un file scan de la tabla.

   b) Estime la cardinalidad del resultado de la selección en términos de cantidad de filas.

   Considere para sus cálculos la siguiente información de catálogo:

Clientes	**Histograma**	
n(Clientes) = 1.200.000	Mexico	250.000
B(Clientes) = 400.000	Argentina	200.000
V(pais, Clientes) = 83	India	150.000

2. *(Procesamiento de consultas)* La plataforma de envios *Lentti* guarda la información de los pedidos de sus usuarios en las siguientes tablas:

   - `Usuarios(id_usuario, email, direccion, ultimo_login)`
   - `Pedidos(id_pedido, id_usuario, fecha, tipo)`

   Por un problema de auditoria, precisa saber en qué fechas un usuario hizo pedidos urgentes con la siguiente consulta SQL:

```
SELECT fecha FROM Usuarios u, Pedidos p
WHERE p.id_usuario = u.id_usuario
AND u.email = 'immcnabb@nfl.com' AND p.tipo = 'URGENTE';
```

   La tabla de usuarios cuenta con un índice por id_usuario (I1) y otro por email (I2). La tabla de pedidos cuenta con un índice por id_usuario (I3) y otro por tipo (I4). Todos son índices secundarios.

   Se pide:

   a) Genere un árbol de consulta para una resolucion eficiente de la consulta.

   b) Calcule el costo de resolver la consulta con el plan que surge de dicho árbol de consulta.

   Para resolver ambos items se cuenta con la siguiente metadata:

Usuarios	**Pedidos**
n(Usuarios) = 80.000	n(Pedidos) = 320.000
B(Usuarios) = 2.000	B(Pedidos) = 32.000
V(email, Usuarios) = 80.000	V(id_usuario, Pedidos) = 80.000
	V(tipo, Pedidos) = 10
Height(I1) = 4	Height(I3) = 4
Height(I2) = 4	Height(I4) = 1

3. *(NoSQL - MongoDB)* El sitio de publicaciones cientificas *Paper View* guarda en una base de datos Mongo los datos de los papers publicados con la siguiente estructura de documento:

```
{
  "_id": 10910355903998401931,
  "titulo": "Base de Datos, de la B a la D",
  "autores": ["Mariano Villani", "Alejandro John"],
  "categoria": "Informatica",
  "puntaje": 4.2
}
```

Lo que buscan es obtener información sobre los autores de papers que pertenezcan a la **categoría "Informática"**: para cada autor que haya publicado al menos 10 de esos papers, quieren conocer la cantidad de esos papers publicados y el promedio de puntaje entre ellos, con la siguiente estructura :

```
{
  "autor": "Mariano Villani",
  "cantidad": 30,
  "promedio_puntaje": 5.3
}
```

   a) Escriba una consulta en MongoDB que devuelva el listado según las condiciones indicadas.

   b) Explique por qué atributos puede shardearse la colección de papers para que la resolución de la consulta sea lo más distribuida posible. En caso de que haya atributos por los que shardear haga la resolución menos distribuida, indique cuales son con una breve explicación del por qué.

4. *(Neo4j)* La famosa red social *LinkedOut* está sufriendo un ataque de trolls! Ha detectado que muchos usuarios se estan organizando para poner puntajes altos a ciertas publicaciones. La información la tiene almacenada en una base de datos en Neo4j con los siguientes nodos y aristas:

```
    (us1:Usuario {username: 'conejo'})
    (us2:Usuario {username: 'aguantemessi'})
    (pub1:Publicacion {titulo: 'Developer SSSSSr', id: '7097321', contenido:
       'Se busca estudiante avanzado....'})
    (pub2:Publicacion {titulo: 'Scrum Master', id: '4032123', contenido: '
       Reconocida empresa busca....'})
    ...
    (us1)-[:PUNTUA {puntaje:10}]->(pub1)
    (us2)-[:PUNTUA {puntaje:9}]->(pub1)
    (us1)-[:PUNTUA {puntaje:10}]->(pub2)
    (us2)-[:PUNTUA {puntaje:10}]->(pub2)
```

Para detectar un par de trolls, busca que ambos hayan puntuado con un puntaje de 8 o mas a al menos 5 publicaciones. Además es necesario que no haya una publicación en la que uno dio un puntaje de 8 o más y el otro la haya puntuado con un 7 o menos. Escriba una consulta en Cypher (lenguaje de consulta de Neo4j) que devuelva, sin repetir, los pares de usuario que cumplan con esas condiciones anteriores.

5. *(Concurrencia)* Dado el siguiente solapamiento de transacciones:

$$\mathtt{b}_{T_1}; \mathtt{b}_{T_2}; \mathtt{b}_{T_3}; \mathtt{W}_{T_1}(\mathtt{X}); \mathtt{R}_{T_3}(\mathtt{X}); \mathtt{R}_{T_2}(\mathtt{Y}); \mathtt{R}_{T_2}(\mathtt{Z}); \mathtt{R}_{T_1}(\mathtt{Z}); \mathtt{c}_{T_1}; \mathtt{R}_{T_2}(\mathtt{X}); \mathtt{W}_{T_3}(\mathtt{Y}); \mathtt{c}_{T_2}; \mathtt{c}_{T_3}$$

   a) Dibuje el grafo de precedencias del solapamiento.

   b) Indique si el solapamiento es serializable. Justifique su respuesta.

   c) Indique si el solapamiento es recuperable. Justifique su respuesta.

6. *(Recuperación)* Un SGBD implementa el algoritmo de recuperación REDO con checkpoint activo. Luego de una falla, el sistema encuentra el siguiente archivo de log:

```
01 (BEGIN, T1);                  09 (BEGIN, T4);
02 (WRITE, T1, A, 10);           10 (COMMIT, T3);
03 (BEGIN, T2);                  11 (WRITE, T4, C, 17);
04 (WRITE, T2, D, 20);           12 (END CKPT);
05 (BEGIN, T3);                  13 (COMMIT, T1);
06 (COMMIT, T2);                 14 (BEGIN CKPT, T4);
07 (BEGIN CKPT, T1, T3);         15 (WRITE, T4, B, 30);
08 (WRITE, T3, B, 40);           16 (COMMIT T4);
```

   Explique cómo se llevará a cabo el procedimiento de recuperación, indicando:

   - Hasta qué punto del archivo de log se deberá retroceder.
   - Qué cambios deberán ser realizados en disco y en el archivo de log.