Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 10 de julio de 2019

TEMA 20191C2					
CRT		**DR**		**Proc.**	
CyT		**NoSQL**		**NoSQL2**	
Corrigió: **Nota:**					

Padrón: _____

Apellido: _____

Nombre: _____

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. Se aprueba con nota mayor o igual a 4(cuatro), equivalente a desarrollar el 60 % del examen correctamente.

1. *(CRT)* La red social *Photogram* almacena las siguientes tablas con información sobre sus usuarios y *posts*, y los *likes* que los usuarios realizan sobre contenido posteado por otros usuarios a los que siguen.

   - `Usuarios(id_usuario, nombre, fecha_alta)`
   - `Posts(id_post, id_usuario, fecha, foto)`
   - `Likes(id_usuario, id_post)`
   - `Seguidores(id_usuario_seguidor, id_usuario_seguido)`

   El usuario cuyo id es `lukin38` solicitó que eliminen a los *stalkers* de su cuenta, definiendo como *stalkers* a aquellas personas que han dado *like* a todas sus publicaciones del último mes. Escriba una consulta en *Cálculo Relacional de Túplas* que permita encontrar el id y el nombre de los *stalkers* de dicho usuario.

2. *(Diseño relacional)* Dada la relación $R(A, B, C, D, E)$ con un conjunto de dependencias funcionales asociado $F = \{A \to B, A \to C, BC \to E, D \to C, E \to D\}$, indique si las siguientes afirmaciones son verdaderas ó falsas, justificando cada una de sus respuestas.

   a) AD es clave candidata de R.

   b) F es un cubrimiento minimal de sí mismo.

   c) La descomposición $\rho(R_1(A, B, C), R_2(B, C, E), R_3(C, D))$ es una descomposición sin pérdida de información.

   d) BE es superclave de R.

   e) $(BE \to C) \in F^+$

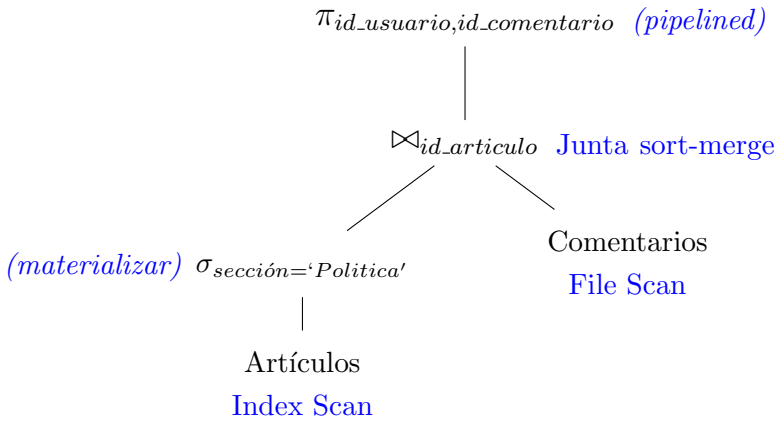
   Nota: La notación F^+ denota la clausura del conjunto de dependencias.

3. *(Procesamiento de Consultas)* Para fortalecer el vínculo con sus lectores, el período *La República Digital* quiere identificar a aquellos usuarios que más comentan sobre política a los efectos de ofrecerles una suscripción de prueba al nuevo suplemento sobre las elecciones de este año. Para ello cuenta con las siguientes dos tablas que almacenan información sobre los artículos periodísticos y los comentarios de los usuarios que han recibido más de 5 *likes*:

   - `Artículos(id_artículo, sección, texto, fecha_publicación)`
   - `Comentarios(id_comentario, id_usuario, id_artículo, texto, fecha_publicación)`

   Adicionalmente, se dispone de un índice de *clustering* por el atributo `sección` para acceder a la tabla `Artículos`.

   El siguiente plan de ejecución busca los pares *(id_usuario, id_comentario)* correspondientes a todos los comentarios escritos por usuarios en artículos de la sección *'Política'*:



k	2^k
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576

   Se pide:

   a) Asumiendo que dispone de una cantidad mínima de memoria de `M=3` (2 bloques para lectura y 1 para escritura), estime el costo del plan de ejecución en términos de cantidad de bloques de disco transferidos.

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

   c) Proponga un índice sobre la tabla `Comentarios` que permita armar un plan de ejecución de menor costo.

   Nota: No es necesario que indique cómo sería dicho plan de ejecución.

   d) *(Bonus track)* Vuelva a estimar el costo del plan de ejecución, asumiendo ahora que dispone de `M=5` bloques de memoria (4 para lectura y 1 para escritura). Explique cómo escala la cantidad de etapas del *sort* externo con la cantidad de memoria disponible.

   Considere para sus cálculos la siguiente información de catálogo:

ARTÍCULOS	**COMENTARIOS**
n(Artículos) = 300.000	n(Comentarios) = 500.000
B(Artículos) = 100.000	B(Comentarios) = 50.000
V(sección, Artículos) = 10	
H(I(sección, Artículos)) = 1	

4. *(Concurrencia y transacciones)* Dado el siguiente par de transacciones:

$$T_1 : \mathtt{b}_{\mathtt{T}_1}; \mathtt{R}_{\mathtt{T}_1}(\mathtt{X}); \mathtt{R}_{\mathtt{T}_1}(\mathtt{Y}); \mathtt{W}_{\mathtt{T}_1}(\mathtt{Z}); \mathtt{c}_{\mathtt{T}_1};$$
$$T_2 : \mathtt{b}_{\mathtt{T}_2}; \mathtt{R}_{\mathtt{T}_2}(\mathtt{Z}); \mathtt{R}_{\mathtt{T}_2}(\mathtt{Y}); \mathtt{W}_{\mathtt{T}_2}(\mathtt{X}); \mathtt{c}_{\mathtt{T}_2};$$

   Se pide:

   a) Proponga un solapamiento de estas dos transacciones que no sea serializable.

   b) Coloque *locks* de lectura y escritura a ambas transacciones de acuerdo con el *Protocolo de Lock de 2 Fases (2PL)*.

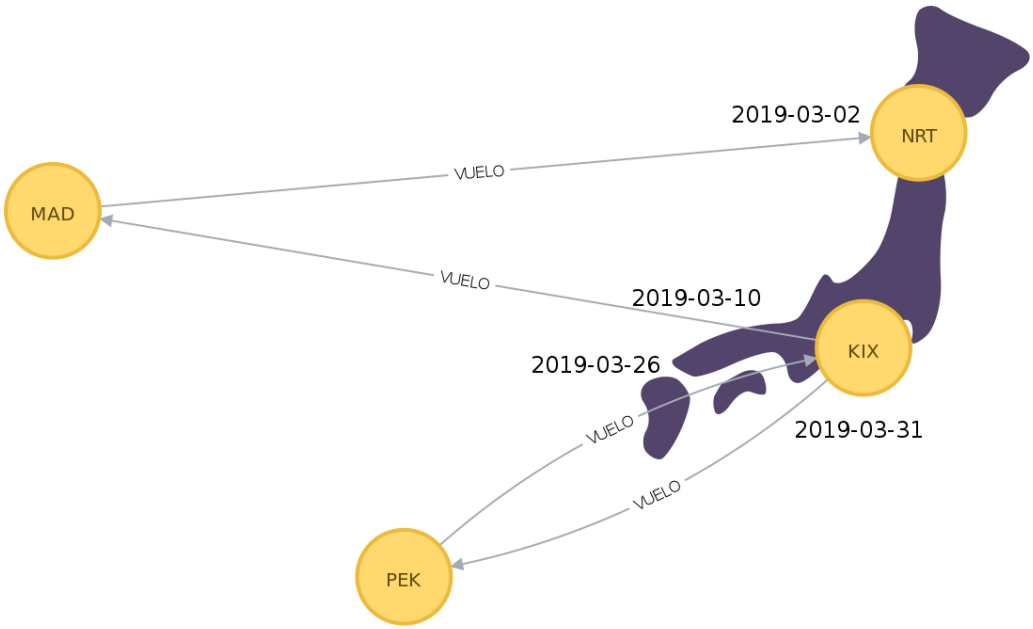
   c) El protocolo *2PL* evitará que se produzca un solapamiento como el que propuso en *a)*, garantizando la serializabilidad. Sin embargo, el uso de este protocolo da lugar a potenciales situaciones de *deadlock*. Muestre un solapamiento de las nuevas transacciones (es decir, las que incluyen las operaciones de *lockeo*) en que se llegue a un *deadlock*. Si considera que ésto no es posible, justifique por qué.

5. *(NoSQL)* Explique el modo en que *MongoDB* gestiona la fragmentación y la replicación a través del *sharding*. Para la fragmentación, describa brevemente cómo se realiza la misma. Para la replicación, indique cómo se realizan las lecturas y escrituras, qué garantías de consistencia se ofrecen, y qué sucede en el caso de que una réplica se caiga.

6. *(NoSQL II)* A Elisa le gusta viajar, y ha creado una base de datos en *Neo4j* para registrar todos sus despegues y aterrizajes en aeropuertos del mundo. En esta base de datos definió a los nodos como aeropuertos y a las aristas como vuelos que ella tomó. Cada aeropuerto se identifica con su código IATA de 3 letras y almacena su nombre y país al que pertenece, mientras que cada vuelo tomado por Elisa se identifica con un código de vuelo y la fecha de salida, almacenando también la fecha de llegada:

```
CREATE (a:Aeropuerto {codigo_IATA: 'EZE', país: 'Argentina',
                      nombre: 'Aeropuerto Internacional de Ezeiza'})
```

```
MATCH (a1: Aeropuerto {código_IATA: 'EZE'}),
      (a2: Aeropuerto {código_IATA: 'CDG'})
CREATE (a1)-[:VUELO {código: 'AF228',
                     fecha_salida: datetime('2019-02-18T12:40:00'),
                     fecha_llegada: datetime('2019-02-19T08:25:00')}]
           -(a2)
```



El país favorito de Elisa es Japón. De hecho, realizando la siguiente consulta, podemos verificar que ingresó a Japón 8 veces:

```
MATCH llegada_japón=(a1: Aeropuerto)-[:VUELO]->(a2:Aeropuerto)
WHERE a1.país<>'Japón' AND a2.país='Japón'
RETURN COUNT(llegada_japón)
```

Elisa quisiera tener un mayor detalle de sus estadías. Escriba una consulta en *Cypher* que devuelva las distintas estadías que Elisa realizó en Japón, indicando para cada estadía la fecha de llegada y la fecha de salida de Japón, ordenando las estadías por fecha de llegada. Para un ejemplo parcial como el de la figura anterior, el resultado debería ser:

fecha_llegada_Japón	fecha_salida_Japón
2019-03-02T12:30:00	2019-03-10T21:15:00
2019-03-26T08:55:00	2019-03-31T13:45:00

Notas: Suponga que Elisa nunca entró ni salió de Japón por un medio que no fuera el transporte aéreo, y que su base de datos se encuentra en un estado consistente, teniendo almacenados todos los viajes que realizó.