Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 12 de diciembre de 2018

TEMA 20182C1					
Proc.I		**Proc.II**		**CyT**	
Seg.		**Esp.**		**NoSQL**	
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. *(Procesamiento de consultas)* Llega Navidad y *Papá Noel* está organizando la repartición de regalos. En breve enviará un *convoy* rumbo a Argentina y necesita construir un listado con todos los regalos a cargar. Para ello cuenta con las siguientes tablas que contienen los datos de todos los niños y niñas del mundo que le hicieron llegar sus cartas, el/los regalo/s que cada uno solicitó y una descripción breve de cada regalo:

   - `Cartas(id_carta, nombre, domicilio, código_postal, país)`
   - `Deseos(id_carta, cod_regalo)`
   - `Regalos(cod_regalo, descripción, precio_rupias)`

   Se pide:

   a) Construya un plan de consulta en forma de árbol que le permita a *Papá Noel* encontrar el listado de códigos de los regalos que debe llevar a la Argentina (sin eliminar los códigos de regalo duplicados).

   b) Proponga dos índices que permitan optimizar el costo de esta consulta, y muestre cómo queda el plan de físico de ejecución sobre el mismo árbol de consulta construído.

   c) Calcule el costo del plan propuesto en términos de cantidad de bloques leídos de disco, considerando que cuenta con un mínimo de memoria disponible y utilizando la siguiente información de catálogo:

CARTAS	DESEOS	REGALOS
n(Cartas) = 400.000.000	n(Deseos) = 800.000.000	n(Regalos) = 30.000
B(Cartas) = 40.000.000	B(Deseos) = 40.000.000	B(Regalos) = 3.000
V(país, Cartas) = 200	V(id_carta, Deseos) = 400.000.000	
	V(cod_regalo, Deseos) = 30.000	

   Nota: Asuma que los índices que definió son de tipo árbol y poseen una altura de 4.

2. *(Procesamiento de Consultas)* Dadas dos relaciones $R(A, B)$ y $S(B, C)$, y la siguiente igualdad entre expresiones del álgebra relacional:

$$\pi_B \left(\sigma_{(A=3)\vee(C=5)}(R * S)\right) = \pi_B \left(\sigma_{A=3}(R) * \sigma_{C=5}(S)\right)$$

Analice si dicha igualdad es una identidad, es decir, si constituye una regla de equivalencia entre expresiones, independientemente de las instancias de R y de S.

a) En caso afirmativo, traduzca la consulta en álgebra relacional al lenguaje del *Cálculo Relacional de Tuplas (C.R.T.)*.

b) En caso negativo, muestre instancias concretas de R y de S para las cuales la igualdad anterior sea falsa.

3. *(Concurrencia y transacciones)* Considere el siguiente solapamiento *serializable* de transacciones:

Transacción T_1	Transacción T_2	Transacción T_3
begin		
		begin
		leer_item(X)
leer_item(Y)		
escribir_item(Y)		
commit		
	begin	
	leer_item(Y)	
		leer_item(Z)
		escribir_item(X)
	leer_item(X)	
	escribir_item(Y)	
		commit
	commit	

Responda los siguientes ítems, justificando cada respuesta.

a) Indique si el solapamiento es recuperable.

b) Indique si el solapamiento evita *rollbacks* en cascada.

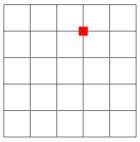
c) Indique si es posible que este solapamiento se haya producido en el contexto de la aplicación del *Protocolo de Lock de 2 Fases Riguroso (R2PL)*.

Nota: Para el punto *c)*, considere el uso de *locks* genéricos. Es decir, no diferencie entre *locks* de lectura y de escritura.

4. *(Seguridad)* Describa el modelo de seguridad *RBAC (Role-Based Access Control)* indicando cuáles son las entidades que lo componen y cómo se interrelacionan entre ellas en dicho modelo. Explique en qué consiste un `GRANT` en SQL, y qué condiciones debe cumplir el usuario que lo ejecuta para que pueda ser otorgado.

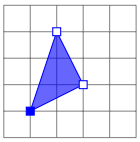
5. *(Bases de datos espaciales)* Juan estaba de paseo por Mataderos y quería comer una pizza en la mejor pizzería del barrio. Para encontrarla se conectó a una base de datos que él mismo había creado en *Postgres/PostGIS* de la siguiente forma:

```
CREATE TABLE Restaurantes (
    cod_restaurante INT PRIMARY KEY,
    nombre VARCHAR(30),
    ubicación GEOMETRY(POINT),
    puntaje FLOAT
  );
```



```
CREATE TABLE Tipos (
    cod_restaurante INT,
    tipo VARCHAR(20),
    PRIMARY KEY (cod_restaurante, tipo),
    FOREIGN KEY (cod_restaurante) REFERENCES Restaurantes(cod_restaurante)
  );
```

```
CREATE TABLE Barrios (
  nombre_barrio VARCHAR(30) PRIMARY KEY,
  límites GEOMETRY(POLYGON)
);
```



El atributo *ubicación* de la tabla `Restaurantes` almacena objetos geométricos de tipo `Point` que representan la ubicación del restaurante, mientras que el atributo *límites* de la tabla `Barrios` almacena objetos de tipo `Polygon` que representan los límites del barrio.

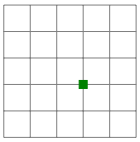
Para resolver el problema, Juan ejecutó la siguiente consulta:

```
SELECT r.nombre, ST_X(r.ubicación) AS longitud, ST_Y(r.ubicación) AS latitud
FROM Restaurantes r, Barrios b, Tipos t
WHERE ST_Contains(b.límites, r.ubicación)
AND t.cod_restaurante = r.cod_restaurante
AND t.tipo = 'Pizzería'
AND b.nombre_barrio = 'Mataderos'
AND puntaje = (SELECT MAX(puntaje)
               FROM Restaurantes r2, Barrios b2, Tipos t2
               WHERE ST_Contains(b2.límites, r2.ubicación)
               AND t2.cod_restaurante = r2.cod_restaurante
               AND t2.tipo = 'Pizzería'
               AND b2.nombre_barrio = 'Mataderos');
```

El resultado fue exitoso y devolvió una única fila. Con las coordenadas devueltas Juan se dirigió a *El Cedrón* y comió la mejor pizza de Mataderos.

Ahora Juan quiere utilizar esta solución para recomendar restaurantes a sus amigos esta misma noche. Sin embargo, los amigos de Juan sólo quieren ir al restaurante más cercano al punto en que se encuentran, independientemente de su puntaje, tipo y barrio. Para esto, Juan creó una nueva tabla *Amigos* de la siguiente forma:

```
CREATE TABLE Amigos (
  nombre VARCHAR(30) PRIMARY KEY,
  ubicación GEOMETRY(POINT)
);
```



Escriba una consulta SQL que encuentre para cada amigo de Juan el nombre de el/los restaurante/s más cercano/s a su ubicación, mostrando el nombre del amigo, el nombre del restaurante y sus coordenadas, y la distancia al punto en que se encuentra dicho amigo.

Nota: Puede utilizar la función `ST_Distance(g1, g2)` incluída en el estándar SQL/MM, que calcula la distancia geodésica entre dos geometrías.

6. *(NoSQL)* Indique si las siguientes afirmaciones relativas a bases de datos distribuidas son verdaderas o falsas, justificando su respuesta.

   a) En una base de datos distribuida en que nunca se producen particiones es posible en teoría obtener máxima consistencia y disponibilidad simultáneamente.

   b) DynamoDB prioriza la consistencia por sobre la disponibilidad.

   c) En Cassandra, el contenido de una *wide row* se almacena entero en un nodo, y puede a la vez estar replicado por entero en varios nodos.

   d) MongoDB permite realizar fragmentación vertical, distribuyendo el contenido de un mismo documento entre distintos nodos.

   e) No es posible implementar transacciones ACID en una base de datos distribuida.