Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2413 - Bases de Datos

Interrogación 2

Fecha: 20 de noviembre de 2024

2º semestre 2024 - Profesores: Eduardo Bustos - Christian Álvarez

---

Si Ud. no lee instrucciones, lea al menos estas INSTRUCCIONES

a) La prueba tiene 4 preguntas, cada una vale 6 puntos (6+1=7) y una pregunta Bonus de 5 décimas sobre la nota final de la prueba.**Duración 2 horas**.

b) Lea toda la prueba primero, las preguntas NO ESTÁN en orden de dificultad

c) Al finalizar la prueba debe digitalizar cada pregunta mas el torpedo en formato PDF VERTICAL y subirlas a Canvas **correctamente**.

d) Cada plana debe contener su Nombre Completo y Número de Lista en la parte superior.

e) No mezcle respuestas de diferentes preguntas en la misma plana. Si puede responder una pregunta en una plana y otra en la otra plana de la misma hoja.

f) Debe firmar la lista de asistencia.

g) No seguir esta instrucción dificulta el trabajo de evaluación y por ende el tiempo de entrega de las notas, además los expone a un descuento de hasta 5 décimas de la nota final.

Pregunta 1 - Lógica en la BD

Usted como experto de BD, ha sido seleccionado para realizar una asesoría de Base de Datos a la empresa DCCEcommerce que se dedica a la venta de artículos por internet. El esquema de la BD es el siguiente:

- Producto (id_prod SERIAL PRIMARY KEY, sku varchar(50), nombre_prod VARCHAR(100), descripcion_prod VARCHAR(500), id_cate INT, activo BOOLEAN, FOREIGN KEY (id_cate) REFERENCES categoria(id_cate))
- Categoria (id_cate SERIAL PRIMARY KEY, nombre_cate VARCHAR(100), descripcion_cate VARCHAR(100), activo BOOLEAN)
- Lista_Precios (id_lista SERIAL PRIMARY KEY, inicio_vigencia DATE, fin_vigencia DATE, tipo INT)
- Producto_Lista (id_prod INT, id_lista INT, precio INT, PRIMARY KEY (id_prod, id_lista))

Con respecto al esquema anterior, es necesario tener en cuenta las siguientes consideraciones:

- La tabla de *Producto* contiene la información de todos los productos que han existido en la empresa DCCEcommerce, donde los productos inactivos tienen el campo *activo* en $FALSE$.
- En la tabla *Categoria* se almacenan todas las categorías a las cuales se puede asociar un producto (1 producto solo puede pertenecer a 1 categoría), teniendo el campo *activo* en $FALSE$ para las categorías que ya no son válidas.
- En la tabla *Lista_Precios* están todas las listas de precios que existen en DCCEcommerce y se usan los campos de fechas para determinar si una lista de precios se encuentra vigente o no.
- Finalmente, la tabla *Producto_Lista* permite realizar la asociación entre *Productos* y *Lista_Precios*. Un producto puede pertenecer a 2 listas de precios. Una lista de precios base a la cual pertenecen todos los productos y una lista con la campaña actual. El *tipo* con valor 0 es la lista base y valor 1 es la oferta actual.

Basándose en el enunciado anterior, responda las siguientes preguntas:

a) Cree un Procedimiento Almacenado **STORED PROCEDURE** que reciba como parámetros una fecha y categoría; y retorne el precio vigente de todos los productos en la fecha entregada como parámetro, para la categoría consultada. En caso de que el producto pertenezca a una lista de oferta, debe seleccionar ese precio y en caso contrario, el precio base. (2 pts)

```
CREATE or REPLACE Function precio_vigente(fecha date, id_cate INT)
RETURNS TABLE (id_prod INT, nombre_prod VARCHAR(100), precio INT)
AS $$
```

```
BEGIN
RETURN QUERY EXECUTE '
    SELECT prod.id_prod, prod.nombre_prod, prod_list.precio
    FROM Producto as prod NNER JOIN
        Categoria as categ ON
            prod.id_cate = categ.id_cate
        INNER JOIN Producto_Lista as prod_list ON
            prod.id_prod = prod_list.id_prod
        INNER JOIN Lista_Precios as list_prec ON
            prod_list.id_lista = lis_prec.id_lista
    WHERE categ.id_cate = $1
    AND list_prec.inicio_vigencia >= $2
    AND list_prec.fin_vigencia <= $2
    AND list_prec.tipo = (
        SELECT max(tipo)
        FROM Lista_Precios as lp INNER JOIN
            Producto_Lista as pl ON
                lp.id_lista = pl.id_lista
        WHERE lp.id_lista = list_prec.id_lista
        AND pl.id_prod= prod_list.id_prod)
    )'
    USING id_cate, fecha;
    RETURN;
END
$$ language plpgsql
```

b) Cree un **TRIGGER**, que al momento de insertar un registro en la tabla *Lista_Precios*, cree una tabla con el nombre $Productos\_Sin\_Precio\_YYYYMMDD$ e inserte todos los productos activos, que no pertenezcan a una lista de ofertas ($tipo = 1$). Ud decida que campos debe tener la tabla. (2 pts)
(YYYY es el año actual, MM el mes actual y DD el día de hoy)

```
CREATE TRIGGER prod_sin_precios
AFTER CREATE ON Lista_Precios
FOR EACH ROW
EXECUTE FUNCTION crear_tabla_productos_sin_precio();

CREATE OR REPLACE FUNCTION crear_tabla_productos_sin_precio() RETURNS TRIGG
DECLARE
    fecha_actual TEXT;
    nombre_tabla TEXT;
BEGIN
```

```
    fecha_actual := TO_CHAR(NOW(), 'YYYYMMDD');
    nombre_tabla := CONCAT('Producto_Sin_Precio_',fecha_actual);

    EXECUTE 'CREATE TABLE IF NOT EXISTS ' || nombre_tabla
    || ' AS
     SELECT p.id_prod, p.nombre_prod
     FROM Producto p
     WHERE NOT EXISTS (
         SELECT 1
         FROM Lista_precios lp INNER JOIN
            Producto_Lista pl
            ON lp.id_lista = pl.id_lista
         WHERE  lp.inicio_vigencia <= NOW()
         AND lp.fin_vigencia >= NOW()
         AND pl.id_prod = p.id_prod
     )';
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

c) Cree una vista **VIEW**, que permita calcular el precio promedio de los productos por categoría al día de hoy. Debe usar el precio de la lista de ofertas y en caso de que el producto no tenga, usar la lista base. (2 pts)

```
CREATE VIEW Precio_Promedio AS
SELECT categ.id_cate, cate.nombre_cate, AVG(prod_list.precio)
    FROM Producto as prod NNER JOIN
        Categoria as categ ON
            prod.id_cate = categ.id_cate
        INNER JOIN Producto_Lista as prod_list ON
            prod.id_prod = prod_list.id_prod
        INNER JOIN Lista_Precios as list_prec ON
            prod_list.id_lista = lis_prec.id_lista
    WHERE list_prec.inicio_vigencia >= NOW()
    AND list_prec.fin_vigencia <= NOW()
    AND list_prec.tipo = (
        SELECT max(tipo)
        FROM Lista_Precios as lp INNER JOIN
            Producto_Lista as pl ON
                lp.id_lista = pl.id_lista
        WHERE lp.id_lista = list_prec.id_lista
        AND pl.id_prod= prod_list.id_prod)
```

```
)
GROUP BY categ.id_cate, categ.nombre_cate
```

Pregunta 2 - ACID, Transacciones, Locks y Recuperación de Fallas

Responda cada una de las siguientes consultas

a) ¿Cuál de las siguientes ejecuciones de transacciones son Conflict Serializables? Para todas las transacciones que no son Conflict Serializables, proponga un nuevo Schedule que sea Conflict Serializable. Justifique sus decisiones.

T1	T2
W(A)	
W(B)	
	W(B)
R(C)	
	W(C)
	R(A)

Cuadro 1: Ejecución 1

T1	T2	T3
	R(Z)	
	R(Y)	
	W(Y)	
		R(Y)
		R(Z)
R(X)		
W(X)		
		W(Y)
		W(Z)
R(Y)		
W(Y)		
	R(X)	
	W(X)	

Cuadro 2: Ejecución 2

T1	T2	T3
		R(Y)
		R(Z)
R(X)		
W(X)		
		W(Y)
		W(Z)
	R(Z)	
R(Y)		
W(Y)		
	R(Y)	
	W(Y)	
	R(X)	
	W(X)	

Cuadro 3: Ejecución 3

Solución:

- La ejecución 1 si es Conflict Serializable. La operación $W(B)$ de $T2$ se puede permutar con la operación $R(C)$ de $T1$, ya que son sobre objetos distintos.
- La ejecución 2 no es Conflict Serializable. La $T2$ depende de la $T1$ ya que ambos usan los objetos X e Y con operaciones de lectura y escritura. Además $T2$ tiene dependencia con la $T3$ que usa el objeto Y, ambos con operaciones de lectura y escritura. Por lo tanto no se pueden realizar permutaciones, para generar una ejecución serializable
- La ejecución 3 si es Conflict Serializable. Las operaciones $R(X$ y $W(X)$ de $T1$ se pueden permutar con las operaciones $W(Y)$ y $W(Z)$ de $T3$, porque son sobre objetos distintos. La operación $R(Z)$ de $T2$ es permutable con las operaciones $R(Y)$ y $W(Y)$ de $T1$ porque son sobre objetos distintos. Por lo tanto la ejecución equivalente sería $T3$, $T1$ y $T2$

b) Para cada una de las transacciones de la pregunta anterior, proponga una ejecución de 2PL. Use como notación $S(A)$ para un lock compartido (shared lock) y $X(A)$ para un lock exclusivo (exclusive lock) del objeto A.

T1	T2
W(A) – X(A)	
W(B) – X(B)	
	W(B) – X(B)
R(C) – S(C)	
	W(C) – X(C)
	R(A) – S(A)

Cuadro 4: Ejecución 1

T1	T2	T3
	R(Z) – S(Z)	
	R(Y) – X(Y)	
	W(Y) –	
		R(Y) – X(Y)
		R(Z) – X(Z)
R(X) – X(X)		
W(X) –		
		W(Y) –
		W(Z) –
R(Y) – X(Y)		
W(Y) –		
	R(X) – X(X)	
	W(X) –	

Cuadro 5: Ejecución 2

T1	T2	T3
		R(Y) – X(Y)
		R(Z) – X(Z)
R(X) – X(X)		
W(X) –		
		W(Y) –
		W(Z) –
	R(Z) – S(Z)	
R(Y) – X(Y)		
W(Y) –		
	R(Y) – X(Y)	
	W(Y) –	
	R(X) – X(X)	
	W(X) –	

Cuadro 6: Ejecución 3

c) Data la ejecución en el Cuadro 7 de una transacción, realice lo siguiente

a) Escriba el log que se genera usando *UNDO LOGGING*

```
Suponemos los siguientes valores iniciales
A = 0, B = 5, X = 10
<START T1>
<T1, A, 0>
<T1, X, 10>
<T1, B, 5>
```

```
<COMMIT T1>
```

b) Escriba el log que se genera usando *REDO LOGGING*

```
Suponemos los siguientes valores iniciales
A = 0, B = 5, X = 10
<START T1>
<T1, A, 10>
<T1, X, 10>
<T1, B, 0>
<COMMIT T1>
<T1 END>
```

T1
T1 START
R(A)
A = A + 10
W(A)
W(X)
R(B)
B = B - 5
W(B)
T1 COMMIT

Cuadro 7: Ejecución Recuperación

c) Describa las acciones que se realizarían usando $UNDO\ LOGGIN$ si ocurre una falla antes de ejecutar $W(B)$

El log se lee desde abajo hacia arriba. Cómo el error es antes de $W(B)$ y no se ha ejecutado $COMMIT$, se realizan las siguientes acciones:

- Escribir 10 en X
- Escribir 0 en A

d) Describa las acciones que se realizarían usando $REDO\ LOGGING$ si ocurre una falla antes de ejecutar $W(B)$

El log se lee desde el principio hasta el final. Cómo el error se produce antes de hacer $COMMIT$, ninguna acción se realiza.

Pregunta 3 - EEDD, Almacenamiento, Índices y Algoritmos

Sea la relación $S(a, b, c, d, e)$ cuyo tamaño es de 10 millones de tuplas, en que cada página de disco contiene P tuplas. Las tuplas de S están distibuídas de manera aleatoria. El atributo c es además una llave candidata, cuyos valores únicos van del 0 al 9.999.999 (distribuidos uniformemente). Para cada una de las consultas a continuación, indique el número de operaciones de I/O que se realizarán para cada una de las siguientes consultas:

a) Si no existe índice: Encontrar todas las tuplas de S tal que $50 < c < 150$.

   Solución: cómo no existe índice, es necesario recorrer todas las páginas para encontrar los registros que nos interesan. Dado que son 10^7 tuplas y en cada página hay P tuplas, el costo es: $\frac{10^7}{P}$.

b) Si existe un índice CLUSTERED tipo B+ Tree sobre el atributo c, el árbol es de altura h y cada página de las hojas está ocupada al 60 %: Encontrar todas las tuplas de S tal que $c < 50$

   Solución: Cómo se necesitan las tuplas donde $c < 50$, las tuplas necesarias son 50 (índice comienza en 0). Cómo las hojas de las páginas están al 60 %, para guardar todas las tuplas se necesitan $\frac{10^7}{0{,}6*P}$. Como en este caso son 50 registros, para encontrar todas las tuplas se necesitan recorrer $\frac{50}{0{,}6*P}$, pero es necesario restar 1 porque es clustered. Para llegar a las hojas el costo es h, por lo tanto el valor es: $h - 1 + \frac{50}{0{,}6*P}$

c) Si existe un índice UNCLUSTERED de tipo Hash con 10 millones de buckets. Cada página del índice contiene M punteros ($M > P$): Encontrar todas las tuplas de S tal que $c = 125$.

   Solución: En este caso cómo necesitamos un sólo registro se necesitan 2 operaciones, una operación para llegar al bucket del índice, dónde se almacenan los índices por ser UNCLUSTERED y la otra para acceder a la página dónde está la tupla.

d) Si existe un índice B+ Tree UNCLUSTERED sobre el atributo c, el árbol es de altura h y cada página contiene M punteros ($M > P$): Encontrar todas las tuplas de S tal que $c < 100$

   Solución: Cómo se necesitan las tuplas donde $c < 100$, las tuplas necesarias son 100 (índice comienza en 0). Como en cáda página hoja se almacenan M punteros, se necesitan $\frac{10^7}{M}$ páginas para todos los punteros. Como en este caso son 100 registros, para todos los punteros se necesitan recorrer $\frac{100}{M}$ páginas. Para llegar a las hojas el costo es h, pero como al recorrer se llega a la primera página con índices, es necesario restar 1. Adicionalmente, hay que ir desde cada puntero a la página por cada tupla, por lo tanto se deben sumar 100. El costo total es: $h - 1 + \frac{100}{M} + 100$

Pregunta 4 - ORM y NoSQL

Responda las siguientes preguntas

a) De la siguiente lista, indique cuáles de las siguientes afirmaciones son Verdaderas o Falsas para las BD Documentales

   a) Verdadero Son flexibles, permitiendo almacenar documentos con distintas estructuras

   b) Verdadero Están orientadas a operaciones rápidas de lectura

   c) Falso Las operaciones de *Join* son más eficientes y rápidas que en SQL

   d) Falso Son ideales para mantener la consistencia de los datos

   e) Falso Son también conocidas como BD Transaccionales

b) Explique el significado de cada una de las siguientes operaciones en Mongo DB

   a) *show collections*: Muestra todas las colecciones en la BD

   b) $db.series.find()$: Todos los documentos de la colección *series*

   c) $db.series.find("type" : "Scripted")$: Todos los documentos de la colección series, dónde la propiedad *type* tiene el valor "*Scripted*"

   d) $db.series.find("runtime" : \$lt : 70)$: Todos los documentos de la colección series, dónde la propiedad *runtime* es menor o igual que 70

   e) $db.series.findOne("rating.avg" : null)$: El primer documento de la colección series, dónde la propiedad *avg*, de la propiedad *rating* es *null*

Bonus - 1/2 punto

Asocie el concepto indicado a continuación con las descripciones indicadas a en la hoja de respuesta.

número	**Descripción**
1	Todas las operaciones de la transacción T1 se hacen antes de las de la T2
2	Aislamiento
3	log de Transacción REDO
4	PRIMARY KEY (llave primaria)
5	Página
6	Atomicidad
7	log operación UNDO en transacción
8	External mergesort
9	Registro de checkpoint bien conformado
10	log de Transacción UNDO
11	Registro de checkpoint mal conformado
12	log operación REDO en transacción

5. Bonus

Número	Descripción
4	llave candidata elegida para hacer un índice
1	Schedule serial
5	Espacio de almacenamiento secundario (HDD) que almacena parte de una relación R
6	se ejecutan todas las operaciones de la transacción, o no se ejecuta ninguna
10	BEGIN TRANSACTION op1 op2 op3 COMMIT
7	registro de log de operación: (Transacción,Página de relación[tupla],valor previo)
12	registro de log de operación: (Transacción,Página de relación[tupla],valor futuro(actual))
11	START CKPT (Transacción T1) op1 op2 op3 END CKPT
3	BEGIN TRANSACTION op1 op2 op3 COMMIT END
2	Cada transacción debe ejecutarse como si se estuviese ejecutando sola, de forma aislada.
8	Algoritmo para ordenar tuplas cuando exceden por mucho la RAM
9	START CKPT (Transacción T1) op1 op2 op3 COMMIT END CKPT