2022 2C Promocional

domingo, 25 de junio de 2023 10:53

(Procesamiento de consultas) Considere los siguientes esquemas de relación que almacenan información sobre los resultados de los partidos del mundial de futsal 2021:

- `equipos(cod_eq, nombre, veces_campeon)`
  // ('ARG', 'Argentina', 1)
- `partidos(id_partido, cod_eq1, cod_eq2, goles_eq1, goles_eq2, fecha)`
  // (23, 'ARG', 'USA', 11, 0, 2021-09-14)
- `jugadores(cod_eq, nro_camiseta, nombre)`
  // ('ARG', 7, 'Leandro Cuzzolino')
- `goles(id_partido, nro_gol, cod_eq, nro_camiseta, minuto, cod_eq_gol_favor)`
  // (23, 3, 'ARG', 7, 4, 'ARG')

a) Se quiere encontrar a los jugadores que utilizan la camiseta número 9 en el equipo en que juegan, devolviendo su nombre y su equipo. Para ello se arma el siguiente plan de consulta:

$$\pi_{nombre\_cod\_eq}\ (\sigma_{nro\_camiseta=9}(jugadores))$$

Estime el costo de la consulta en los siguientes dos escenarios, asumiendo que dispone de `M=10` bloques de memoria para el procesamiento:

1) Utilizando un índice secundario por el atributo `nro_camiseta` para la selección.
2) Utilizando un índice de clustering por el atributo `nro_camiseta` para la selección.

En ambos casos, considere que los índices son de tipo árbol y tienen altura 4. Además, considere para sus cálculos la siguiente información de catálogo:

JUGADORES
n(jugadores) = 1000
B(jugadores) = 50
V(cod_equipo, jugadores) = 40
V(nro_camiseta, jugadores) = 30

b) Dadas dos tablas $R(A, B)$ y $S(B, C, D)$, se quiere calcular la junta natural entre ambas. Dado que se dispone de sólo `M=100` bloques de memoria (de manera que ninguna de las dos tablas entra en memoria), y que no se cuenta con índices para la junta, se utilizará el método de junta hash GRACE. A continuación se indican algunas estadísticas de las tablas:

R(A,B)	S(B,C,D)
n(R) = 300.000	n(S) = 50.000
B(R) = 30.000	B(S) = 5.000
V(B, R) = 300	V(B, S) = 500

Se pide:

1) Indique qué cantidad k de particiones intentaría generar para que la junta hash GRACE sea factible de ser realizada, justificando su respuesta. Estime qué tamaño promedio –en términos de cantidad de bloques– tendrían las particiones en ese caso.
2) Bajo esa hipótesis, estime cuál sería el costo de realizar la junta, en términos de cantidad de accesos a bloques de disco.

a)

$n(\sigma) = \frac{n(R)}{V(nro,R)} = \lceil \frac{1000}{30} \rceil = 34$

$F(R) = \frac{1000}{50} = 20$

$B(\sigma) = \lceil \frac{n(\sigma)}{F(R)} \rceil = 2$

↳ Es posible utilizar pipelining para la proyeccion
↳ No hay costo extra
↳ Son PK, siempre unicos

1) $c(\sigma) = H(I(nro,R)) + \lceil \frac{n(R)}{V(nro,R)} \rceil$

$= 4 + 34 = 38$ lecturas de bloque

2) $c(\sigma) = H(I(nro,R)) + \lceil \frac{B(R)}{V(nro,R)} \rceil$

$= 4 + 2 = 6$ lecturas de bloque

b) La cantidad de particiones tiene que ser tal que

- Al menos un bloque de cada particion entre en memoria. $K \leq M$
- Todos los bloques de una particion, para al menos una entre en memoria
  ↳ $K \leq \min(R_i, S_i)$

$F(R) = 10$ $F(S) = 10$

$n(B_i,R) = 1000$ $n(B_i,S) = 100$

300 valores unicos
~100 bloques por cada valor

A lo sumo $K = 300$
↳ 100 bloques por particion

500 valores unicos
~10 bloques por cada valor

A lo sumo $K = 500$
↳ 10 bloques por particion

Pero $K \leq 98$ → (1 para resultado, 1 para "hacer desfilar"

Si $K = 98$

$310 = \lceil 300 \cdot 100 \rceil$ ←

pero $K < 98$ ...

$310 = \lceil \frac{300}{98} \cdot 100 \rceil$ ← Si $K = 98$

bloques por partición

↳ $\lceil \frac{500}{98} \cdot 10 \rceil = 52$ bloques por partición

El costo sera el de hashear ambas tablas $2(B(R) + B(S))$, sumado al costo de comparar $B(S_i)$ con $B(R_i)$, $B(S_i) + B(R_i)$ por cada particion $\Rightarrow B(R) + B(S)$

$\Rightarrow 3(B(R) + B(S))$

2. *(NoSQL)*

a) *(MongoDB)* La FIFA organiza una votación en la que cada participante debe escoger a 3 jugadores candidatos a ser goleadores de la Copa Mundial de Qatar 2022. Los candidatos escogidos por cada usuario se almacenan en un documento como el siguiente de una base de datos en *MongoDB*:

```
1 id_usuario: 53
2 pais_origen: 'ARGENTINA'
3 candidatos: ['Lionel Messi', 'Kylian Mbappé', 'Julián Alvarez']
```

La FIFA desea estimar de qué países provienen los hinchas que han elegido a Lionel Messi entre sus candidatos.

Se pide:

1) Escriba una consulta en MongoDB que devuelva para cada país de origen la cantidad de hinchas que han seleccionado a Lionel Messi entre sus candidatos, ordenando el resultado por dicha cantidad en forma descendente.
2) Suponga que cientos de millones de hinchas han participado de esta votación, y que para agilizar las consultas, se decide shardear la colección distribuyendo la información en 200 nodos. Se debe decidir si la colección será shardeada por id del usuario o por país de origen. ¿Ambas opciones le parecen factibles, o es necesario que la colección sea shardeada por alguno de ellos en particular para que la consulta pueda realizarse? Justifique su respuesta.

aggregate

```
[ { $match : { candidatos : { $all : ['Leonel Messi'] } } }
{ $group : { _id: $pais_origen
             cantidad : { $count : 1 } } }
{ $sort : { _id : -1 } } ]
```

2) Sera mejor shardear por pais de respuesta, utilizando nodos restantes como nodos secundarios ($|pais\_origen| < 200$).

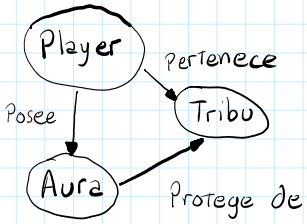
De esta manera consultas dirigidas a algun usuario o pais en particular seran hechas en un solo nodo

Para la consulta (1) en particular cada nodo sera encargado de un solo pais para la agregación. En cambio si se hubiese hecho por id_usuario, todos los nodos tendrian que haber compartido informacion entre si

* Se asume que id_usuario no contiene informacion codificada

b) *(Neo4j)* En el juego online *Prudence* los jugadores arman equipos de manera que cada jugador escoge una única tribu, y debe buscar "auras" que le den protección frente a jugadores de otras tribus. La búsqueda debe ser prudente, porque si el jugador se acerca a menos de 20 pies de un jugador de una tribu frente a la cual no está protegido, es inmediatamente aniquilado. Los desarrolladores del juego mantienen toda esta información en una base de datos en Neo4j, incluyendo la tribu de pertenencia del jugador y las auras que posee:

```
(p:Player {nombre: 'Juan'})
(t:Tribu {nombre: 'Terakee'})
(p)-[:PERTENECE_A]-(t)
...
(a:Aura {nombre: 'pluma_brillante'})
(p)-[:POSEE]-(a)
(a)-[:PROTEGE_DE]-(t2)
```



Escriba una consulta en Neo4j que permita que un jugador de nombre 'x' sepan quienes son los jugadores de los que debe mantenerse alejados, por no poseer un aura que lo protega de ellos.

```
MATCH
  (r:Player)
WHERE NOT EXISTS (f:Player)-[:PERTENECE]-(:Tribu)-[:PERTENECE]-(x:Player {name:'x'})
AND NOT EXISTS (p:Player)-[:PERTENECE]-(:Tribu)-[:PROTEGE_DE]-(:Aura)-[:POSEE]-(x:Player {name:"x"})
AND r.name != "x"
RETURN r
```

3. *(CRT)* Considere las mismas relaciones del ejercicio 1.a), y escriba una expresión en Cálculo Relacional de Tuplas que devuelva las fechas en que no hubo ningún gol en ningún partido.

- `equipos(cod_eq, nombre, veces_campeon)`
  // ('ARG', 'Argentina', 1)
- `partidos(id_partido, cod_eq1, cod_eq2, goles_eq1, goles_eq2, fecha)`
  // (23, 'ARG', 'USA', 11, 0, 2021-09-14)
- `jugadores(cod_eq, nro_camiseta, nombre)`
  // ('ARG', 7, 'Leandro Cuzzolino')
- `goles(id_partido, nro_gol, cod_eq, nro_camiseta, minuto, cod_eq_gol_favor)`
  // (23, 3, 'ARG', 7, 4, 'ARG')

$$\{ p.fecha \mid Partidos(p) \wedge \neg(\exists g)\ (Goles(g) \wedge p.id\_partido = g.id\_partido) \}$$

4. *(Concurrencia y Recuperación)*

a) *(Concurrencia)* Considere el siguiente solapamiento de 3 transacciones:

$$b_{T_1}; b_{T_2}; b_{T_3}; R_{T_1}(X); R_{T_1}(Y); R_{T_2}(Y); R_{T_2}(Z); W_{T_2}(X); R_{T_3}(Z); R_{T_3}(X); W_{T_3}(Z); c_{T_1}; c_{T_2}; c_{T_3}$$

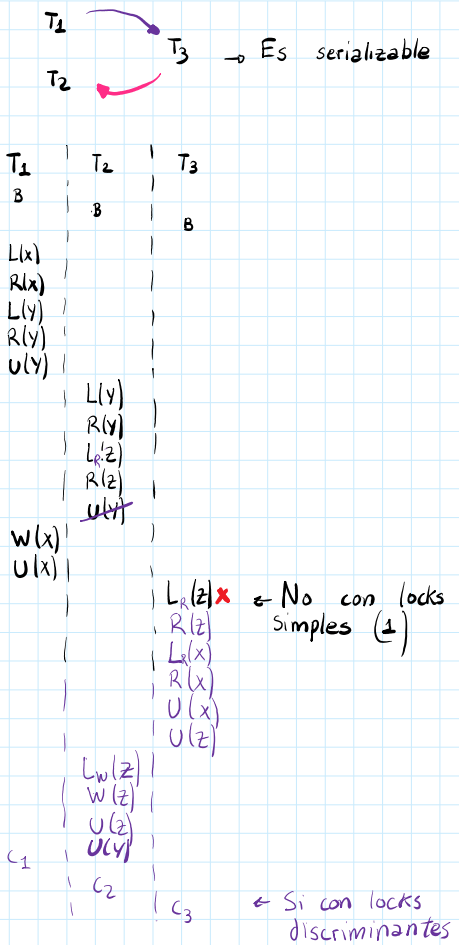
Se pide:

1) Explique si es posible que este solapamiento ocurra utilizando el protocolo de lock de dos fases (2PL). Para ello, intente colocar *locks* `L()` y *unlocks* `U()` respetando el protocolo, y analice si ello es factible.
2) Indique si el solapamiento es serializable, justificando su respuesta.
3) Indique si el solapamiento es recuperable, justificando su respuesta.
4) Indique si es posible que este solapamiento se haya producido utilizando el mecanismo de control de concurrencia *Snapshot Isolation*. Si es posible, entonces indique cuál sería un orden serial equivalente a esta ejecución.

b) *(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);
02 (WRITE, T1, X, 5);
03 (BEGIN, T2);
04 (WRITE, T2, Y, 18);
05 (BEGIN, T3);
06 (WRITE, T2, Z, 12);
07 (COMMIT, T2);
08 (WRITE, T3, Z, 10);
09 (BEGIN CKPT, {T1, T3});
10 (WRITE, T1, Y, 3);
11 (COMMIT, T1);
12 (WRITE, T3, Y, 12);
13 (BEGIN, T4);
14 (WRITE, T4, X, 8);
15 (END CKPT);
16 (COMMIT, T4);
```

Explique cómo se llevará a cabo el procedimiento de recuperación, indicando qué cambios deben ser realizados en disco y en el archivo de log.



← No con locks Simples (1)

← Si con locks discriminantes

Definición: Un solapamiento es recuperable si y sólo si ninguna transacción *T* realiza el *commit* hasta tanto todas las transacciones que escribieron datos antes de que *T* los leyera hayan commiteado.

T_1: Lee independiente a otras trans, puede commitear cuando quiera

T_2: Idem T_1

T_3: Lee X modificado por T_2

T_3 hace commit luego de que T_2 lo haga ⇒ es recuperable

4) No hay conflictos de ww o ciclos RW, por lo que es posible bajo snapshot isolation

REDO

```
01 (BEGIN, T1);
02 (WRITE, T1, X, 5);   Escribir X=5
03 (BEGIN, T2);
04 (WRITE, T2, Y, 18);
05 (BEGIN, T3);
06 (WRITE, T2, Z, 12);
07 (COMMIT, T2);
08 (WRITE, T3, Z, 10);
09 (BEGIN CKPT, {T1, T3});
10 (WRITE, T1, Y, 3);   Escribir Y=3
11 (COMMIT, T1);
12 (WRITE, T3, Y, 12);  Ignorar
13 (BEGIN, T4);
14 (WRITE, T4, X, 8);   Escribir X=8
15 (END CKPT);
16 (COMMIT, T4);
(ABORT, T3)
```

Recuperación

Trans con commit: [T_4, T_2, T_4]

Trans sin commit: [T_3]

↑ Ya en disco por checkpoint no hace falta escribir