Final 5-2-20

Magali Marijuan

Ejercicio 1

Enunciado:

1. *(SQL)* Kobe Bryant ha sido uno de los más grandes basquetbolistas de todos los tiempos. Accederemos a una base de datos de la NBA para extraer algunos datos sobresalientes acerca de su carrera.

   A continuación se muestran las tablas de las que disponemos, y que contienen información sobre los equipos de la NBA, los partidos disputados en cada temporada, los equipos de los que cada jugador formó parte, y las acciones realizadas por cada jugador en cada partido que jugó (cantidad de puntos anotados, minutos jugados, etcétera):

   - `Seasons(name, starting_date, end_date)`
   - `Teams(name, city, conference)`
   - `Matches(match_id, local_team, visiting_team, date, stage_name, season_name)`
   - `Players(name, birth_date, height)`
   - `TeamsCompositions(player_name, season_name, team_name)`
   - `Actions(player_name, match_id, minutes, points, rebounds, steals, blocks)`

   Nota: Por simplicidad se asume que los jugadores no cambian de equipo en medio de la temporada.

   Para cada ítem indicado a continuación, escriba una consulta SQL que permita encontrar la respuesta a partir de la información contenida en las tablas anteriores. Sólo a modo de cultura general, podrá encontrar la respuesta a cada estadística girando la página.

   a) Encuentre la cantidad de partidos en que Kobe Bryant marcó al menos 50 puntos, devolviendo únicamente dicha cantidad.

Rta: Lo hizo en **25** partidos (siendo únicamente superado por Wilt Chamberlain -en 118 ocasiones- y Michael Jordan -en 31 ocasiones-).

b) Kobe Bryant llegó a convertir 5640 puntos en los *playoffs*. Encuentre a aquellos jugadores que hayan superado la marca de 5640 puntos, indicando para cada uno su nombre y la cantidad de puntos que convirtió en *playoffs*.

Nota: Los *playoffs* son una etapa de la temporada. La etapa a la que corresponde cada partido se indica bajo el atributo `stage`.

Rta: Kobe Bryant fue únicamente superado por **LeBron James** (6911 puntos), **Michael Jordan** (5987 puntos) y **Kareem Abdul-Jabbar** (5762 puntos).

c) Encuentre la cantidad de puntos que marcó Kobe Bryant en el último partido oficial que jugó, indicando el nombre del equipo local, el nombre del equipo visitante y la cantidad de puntos marcados por Kobe.

Rta: En su último partido oficial, que enfrentó a **Los Angeles Lakers** contra los **Utah Jazz**, Kobe Bryant señaló **60** puntos (era la séptima vez que alcanzaba dicha marca).

Solución

a)

```
-- equipos en los que jugo por temporada --
WITH AUX1 as(
    SELECT team_name, season_name
    FROM TeamsCompositions
    WHERE player_name like 'KOBE BRYANT'
);

-- partidos que jugo --

WITH AUX2 as(
    SELECT m.match_id
    FROM AUX1 as a1 INNER JOIN Matches as m ON (
        (a1.team_name = local_team OR a1.team_name = visiting_team) AND
        a1.season_name = m.season_name
    ) as aux2
);

-- SOLUCIÓN --
SELECT points
FROM Actions INNER JOIN AUX2 using(match_id) as aux3
WHERE player_name LIKE 'KOBE BRYANT' and points >= 50
```

b)

```
-- partidos de playoff --
WITH AUX1 as(
    SELECT match_id
    FROM Matches
    WHERE stage_name LIKE "playoffs"
);


-- SOLUCIÓN

SELECT player_name, total_play_off_points
FROM (
    SELECT player_name, SUM(points) as total_play_off_points
    FROM Matches INNER JOIN AUX1 USING(match_id)
    GROUP BY player_name
) as AUX2
WHERE total_play_off_points >= 5640

-- o si no

SELECT player_name, SUM(points) as total_play_off_points
FROM Matches INNER JOIN AUX1 USING(match_id)
GROUP BY player_name
HAVING SUM(points)>= 5640
```

c)

```
-- equipos en los que jugó --
WITH AUX1 as(
    SELECT team_name, season_name
    FROM TeamsCompositions
    WHERE player_name like 'KOBE BRYANT'
);

-- ultimo partido que jugó
WITH AUX2 AS (
    SELECT match_id, local_team, visiting_team, max(date)
    FROM Matches INNER JOIN AUX1 (
        (a1.team_name = local_team OR a1.team_name = visiting_team) AND
        a1.season_name = m.season_name
    )
);

-- SOLUCIÓN
SELECT local_team, visiting_team, points
FROM Actions INNER JOIN AUX2 USING(match_id)
WHERE player_team like 'KOBE BRYANT'
```

Ejercicio 2

Enunciado:

2. *(Álgebra Relacional)* Las siguientes tablas almacenan información sobre los precios a los que se venden distintos productos en supermercados de la Ciudad de Buenos Aires.

   - `Supermercados(nombre_super, dirección, cant_empleados)`
   - `Productos(cod_barras, nombre_producto, categoría, descripción)`
   - `Precios(nombre_super, cod_barras, precio)`

   Nota: Si un producto no tiene precio asignado para un supermercado, esto implica que ese supermercado no comercializa dicho producto.

   Para cada una de las siguientes consultas en álgebra relacional exprese en lenguaje coloquial –con precisión– qué es lo que hace:

   a) $\pi_{\texttt{nombre\_super}}(\texttt{Supermercados}) - \pi_{\texttt{nombre\_super}}(\sigma_{\texttt{categoria}=\text{`Frescos'}}(\texttt{Productos} \bowtie \texttt{Precios}))$

   b) $\pi_{\texttt{nombre\_super}}(\texttt{Supermercados}) - \pi_{\texttt{P1.nombre\_super}}(\rho_{\texttt{P1}}(\texttt{Precios}) \bowtie_{(\texttt{P1.cod\_barras}=\texttt{P2.cod\_barras}) \wedge (\texttt{P1.precio}<\texttt{P2.precio})} \rho_{\texttt{P2}}(\texttt{Precios}))$

   c) $\pi_{\texttt{nombre\_super},\texttt{categoria}}(\texttt{Productos} \bowtie \texttt{Precios}) \div \pi_{\texttt{categoria}}(\texttt{Productos})$

Solución

a) Toma `Productos` y `Precios` y hace la junta por el campo `cod_barras` . Se queda todas las tuplas cuya categoría sea `Frescos` Y se queda con el atributo `nombre_super` de cada tupla (eliminando repetidos). Luego a supermercados le quita todos los supermercados que resultaron de tener la categoría frescos y se queda con los nombres de esos super.

En fin, se queda con todos lo supermercados que no vendan `Frescos`

b) Renombra la relación `Precios` como `p2` . Renombra la relación `Precios` como `p1` . Hace la Junta entre `p1` y `p1` siempre que el código de barra de cada tupla de `p1` sea igual a la tupla de `p2` y el precio de `p1` sea menor que el precio de `p2` . Del resultado se queda con el atributo `nombre_super` sacando repetidos y se los resta a `Supermercados` quedandose nuevamente con los nombres de los supers.

En fin, se queda con todos los uspermercados que vendan algún producto al menor precio.

c) Se queda todas las categorías (sin repetidos). Toma `Productos` y `Precios` y hace la junta por el campo `cod_barras` . Se queda con el `nombre_super` y `categoria` de todos los supers que vendan en todas las categorías.

Ejercicio 3

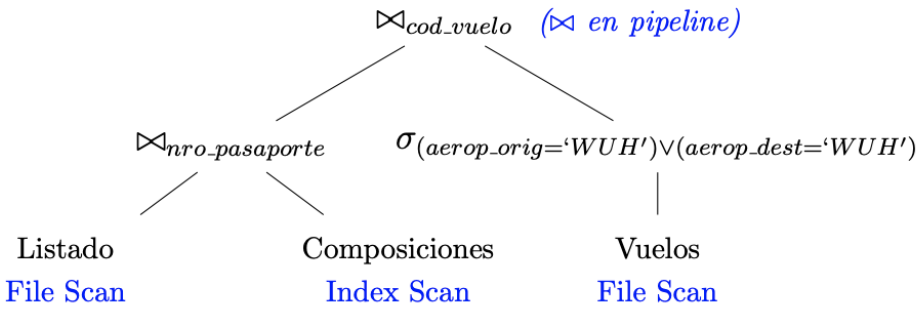
Enunciado:

3. *(Procesamiento de consultas)* Ante la emergencia del *coronavirus* se ha instalado un protocolo de seguridad en todos los aeropuertos del país, pare verificar si alguno de los pasajeros en arribo tomó un vuelo a Wuhan en los últimos 30 días. A estos efectos, para cada vuelo en arribo se dispone del listado de pasajeros, y además se tiene acceso a una tabla con todos los códigos de vuelo registrados en el mundo, y a una tabla global que registra las composiciones históricas de todos los vuelos durante un plazo de 30 días:

   - `Listado(nro_pasaporte, apellido, nombre)`
   - `Composiciones(cod_vuelo, fecha, nro_pasaporte)`
   - `Vuelos(cod_vuelo, aerop_orig, aerop_dest)`

   Adicionalmente, la tabla `Composiciones` posee un índice secundario por el atributo `nro_pasaporte`.

   Acaba de llegar el vuelo AA1135 proveniente de Madrid con 200 pasajeros, y se construyó el siguiente plan de ejecución para saber si alguno de sus pasajeros estuvo en Wuhan en los últimos 30 días:



   Considere la siguiente información de catálogo:

LISTADO	COMPOSICIONES	VUELOS
n(Listado) = 200	n(Composiciones) = 300M	n(Vuelos) = 100.000
B(Listado) = 20	B(Composiciones) = 30M	B(Vuelos) = 10.000
	V(nro_pasaporte, Composiciones) = 100M	V(aerop_orig, Vuelos) = 10.000
	H(I(nro_pasaporte, Composiciones)) = 4	V(aerop_dest, Vuelos) = 10.000

Se pide:

a) Estime el costo del plan de ejecución en términos de cantidad de bloques. Puede asumir que dispone de una cantidad de memoria ilimitada.

b) Si el índice de `Composiciones` por el atributo `nro_pasaporte` fuera un índice de *clustering* en vez de secundario, ¿cree que el costo de la consulta sería menor? Justifique su respuesta.

c) Si la tabla `Vuelos` tuviera un índice primario por el atributo `cod_vuelo`, cree que podría disminuirse el costo de la consulta realizando un *index scan* sobre ella? Justifique su respuesta.

Solución

a)

Utilizo unico loop

$$C(\bowtie_{nro\_pasaporte}) = B(LISTADO) + n(LISTADO)(Height(nro\_pasaporte, COMPOSICIONES) + \frac{n(COMPOSICIONE}{V(nro\_pasaporte, COMPOSI}$$

$$C(\bowtie_{nro\_pasaporte}) = 20 + 200(4 + \frac{300M}{100M}) = 1420$$

No contabilizo el costo de $C(\sigma_{(aerop.orig='WUH')ORaerop.dest='WUH')})$ por pipelining con la junta siguiente.

Utilizo Junta Hash -> porque se cuenta con limitada memoria

$$C(\bowtie_{cod_vuelo}) = 3(C(\bowtie_{nro\_pasaporte}) + C(\sigma_{(aerop.orig='WUH')ORaerop.dest='WUH')}) = 3(C(\bowtie_{nro\_pasaporte}) + B(VUELOS)) = 1420 + 1000$$

b) El costo sería menor porque el costo de index_scan con idx de clustering sería:

$$C(t1) = Height(nro\_pasaporte, COMPOSICIONES) + \frac{B(COMPOSICIONES)}{V(nro\_pasaporte, COMPOSICIONES)} = 4 + (30M/100M) = 4.3$$

Que es menor a 7, lo que habíamos obtenido antes. Esto es porque al ser de clustering los datos en la tabla están ordenados bajo este índice y se puede implementar alguna lógica para traer los bloques que sólo están compuestos (casi todos) por el número de pasaporte buscado. Hay que considerar que los índices de clustering son más costoso de mantener, porque cada vez que se da una inserción hay que ordenar.

c) No, porque para hacer la selección se require ver todas las tuplas y por lo tanto, traer todos lo bloques de memoria. Podría, incluso, ser más costos, ya que involucra traer los bloques del index tree.

Ejercicio 4

Enunciado:

4. *(Concurrencia y transacciones)* En este ejercicio construiremos un ejemplo minimalista para ilustrar de qué manera el uso de *locks* puede evitar problemas de serializabilidad en la ejecución de transacciones. Para ello, diseñe un solapamiento no serializable de transacciones y muestre que utilizando los *locks* bajo ciertas reglas dicho solapamiento no hubiera podido producirse.

Solución

Suponiendo que estamos resolviendo bajo serializabilidad del tipo **por conflicto**.

Para implementar un **control basado en locks** y asegurar la serializabilidad es importante implementar el **protocolo de dos fases**. Estos nos dice que cada transancción debe tomar todos locks (siguiendo un orden asignado a los items) al comienzo de la transacción y liberarlos al final.

Para evitar problemas de **deadlock e inanición**:

- Adquirir todos los locks al principio
- Establecer un órden de los items (podría estar basdo en time stamp)
- Definir un grafo de alocación de recursos y timeout para adquisición de lock.
- Encolar los pedidos de locks de manera tal que las transacciones que esperan de hace más tiempo por un recurso tengan prioridad en la adquisición dle lock.

T1	T2
b	
$R_{T1}(A)$	
-	b
-	$R_{T2}(A)$
-	$R_{T2}(B)$
$W_{T1}(A)$	

T1	T2
$R_{T2}(B)$	
-	$W_{T2}(B)$
c	
-	c

Vemos que hay un conflicto de T2 -> T1 porque ($R_{T2}(A)$, $W_{T1}(A)$) y de T1 -> T2 porque ($R_{T2}(B)$, $W_{T2}(B)$).

Si implementamos el protocolo de dos fases:

- Definimos que el orden para requerir locks sea primero el de A y luego el de B.

T1	T2
b	
$LOCK_{T1}(A)$	
$LOCK_{T1}(B)$	
$R_{T1}(A)$	
$W_{T1}(A)$	
$R_{T2}(B)$	
$UNLOCK_{T1}(A)$	
$UNLOCK_{T1}(B)$	
c	
-	b
-	$LOCK_{T2}(A)$
-	$LOCK_{T2}(B)$
-	$R_{T2}(A)$
-	$R_{T2}(B)$
-	$W_{T2}(B)$
-	$UNLOCK_{T2}(A)$
-	$UNLOCK_{T2}(B)$
-	c

Ejercicio 5

Enunciado:

5. *(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, A, 10);
03 (BEGIN, T2);
04 (WRITE, T2, B, 8);
05 (WRITE, T2, C, 3);
06 (COMMIT, T1);
07 (BEGIN CKPT, T2);
08 (BEGIN, T3);
09 (BEGIN, T4);
10 (WRITE, T3, A, 12);
11 (COMMIT T2);
12 (WRITE, T3, C, 5);
13 (END CKPT);
14 (WRITE, T4, B, 22);
15 (COMMIT, T3);
```

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

Solución

Como en la línea 14 se encuentra con un END CKPT debe retroceder al BEGIN de la transacción activa más antigua definida en la linea 7, que es T2.

Rehacer el write de la línea 4, 5.
Rehacer el write de la línea 10 y 12.

Abortar T4 y volcar el log a disco.

Ejercicio 6

Enunciado:

Solución

6. *(NoSQL)* Explique en qué consiste el concepto de *wide row* utilizado en Cassandra, y qué ventajas presenta respecto al concepto de fila de una tabla en una base de datos relacional.

TODO: completar mañana