Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 5 de febrero de 2020

TEMA 20192C3					
SQL		**AR**		**Proc.**	
CyT		**Rec.**		**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. *(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).

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

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

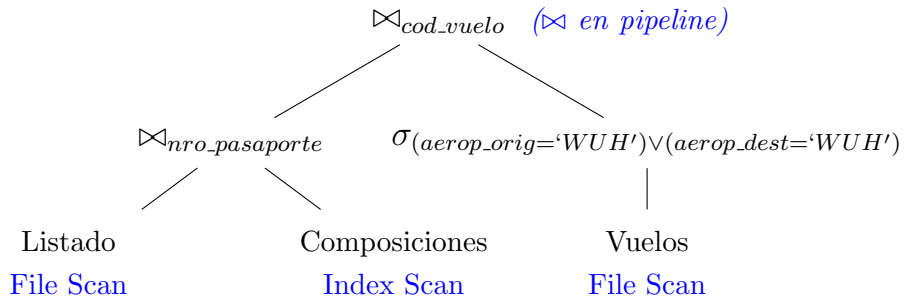
   c) $\pi_{\texttt{nombre\_super,categoria}}(\texttt{Productos} \bowtie \texttt{Precios}) \div \pi_{\texttt{categoria}}(\texttt{Productos})$

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.

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.

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.

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.