Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 3 de julio de 2019

TEMA 20191C1					
SQL		**Proc.**		**DR**	
CyT		**Seg.**		**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)* En el último censo poblacional se recolectó información sobre el nivel educativo alcanzado por las personas, con el objetivo de estudiar la brecha en el nivel educativo entre padres e hijos. La información que queremos analizar se encuentra recogida en las siguientes dos tablas:

   - `Persona(dni, apellido, nombre, fecha_nacimiento, maximo_grado_alcanzado)`
   - `HijoDe(dni_hijo, dni_padre)`

   En donde el máximo grado alcanzado es una variable categórica que se representa con los siguientes valores enteros en orden de grado: `1- SIN INSTRUCCIÓN; 2- PRIMARIO INCOMPLETO; 3- PRIMARIO COMPLETO; 4- SECUNDARIO INCOMPLETO; 5- SECUNDARIO COMPLETO; 6- TERCIARIO/UNIVERSITARIO INCOMPLETO; 7- TERCIARIO/UNIVERSITARIO COMPLETO`.

   La pregunta a responder es en qué medida las nuevas generaciones alcanzan un nivel educativo superior al que alcanzaron sus padres. Para ello, escriba una consulta SQL que encuentre la cantidad de personas nacidas desde 1980 en adelante y de más de 25 años de edad al 27/10/2010 que hayan alcanzado un nivel educativo superior al que alcanzaron sus padres.

   Nota: Considere únicamente a aquellas personas de las cuales al menos uno de sus padres participó del censo. En caso de que ambos padres hubieran participado del censo, se requiere que el nivel educativo de la persona sea superior al de *ambos* padres.

2. *(Procesamiento de Consultas)* En plena Copa América, la CONMEBOL dispone de un sistema de inteligencia artificial que reconoce el instante en que cada jugador toma posesión del balón. Dicho sistema loguea en una tabla `Posesiones` cada instante de cada partido en que un jugador toma el balón, o bien en que el balón queda fuera de juego (en cuyo caso registra un valor nulo). Asímismo cuenta con una tabla `Jugadores` con algunos datos básicos de cada jugador, incluyendo el equipo en que juega:

   - `Posesiones(cod_partido, timestamp, cod_jugador)`
   - `Jugadores(cod_jugador, apellido, nombre, fecha_nac, equipo)`

   A continuación se muestran dos fragmentos de instancias de ejemplo.

POSESIONES

cod_partido	timestamp	cod_jugador
CF-2	16:21:18.418	149
CF-2	16:21:22.637	151
CF-2	16:21:29.827	NULL
CF-2	16:21:42.402	223
CF-2	16:21:46.535	150
CF-2	16:21:49.388	149
CF-2	16:21:55.109	NULL
...	...	...

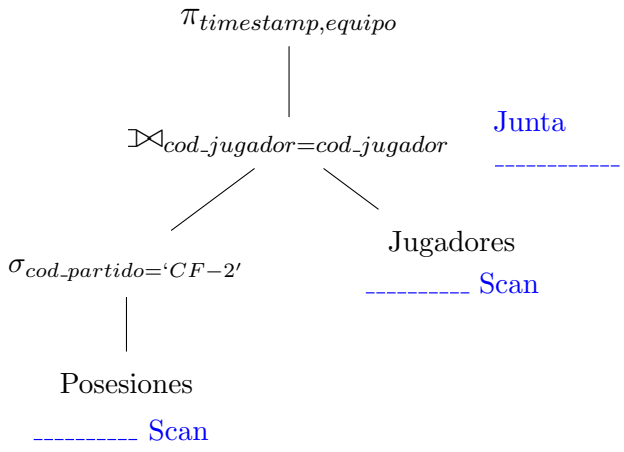
JUGADORES

cod_jugador	apellido	nombre	fecha_nac	equipo
149	Messi	Lionel	24/06/87	Argentina
150	Otamendi	Nicolás	12/02/88	Argentina
151	Lo Celso	Giovanni	09/04/96	Argentina
152	Alves	Dani	06/05/83	Brasil
153	Jesús	Gabriel	03/04/97	Brasil
...	...	...	...	...

   La tabla `Posesiones` posee un índice primario de tipo árbol, y por lo tanto está ordenada por su clave. Este es un índice compuesto que se puede acceder tanto por el par {`cod_partido, timestamp`} como por el atributo `cod_partido`. A continuación se muestra la información de catálogo disponible sobre las tablas:

POSESIONES	**JUGADORES**
n(Posesiones) = 50.000	n(Jugadores) = 360
B(Posesiones) = 5.000	B(Jugadores) = 90
V(cod_partido, Posesiones) = 20	V(equipo, Jugadores) = 12
H(I({cod_partido, timestamp}, Posesiones)) = 4	

   Considere el siguiente árbol de consulta para generar una tabla `Posesiones_ArgVen(timestamp, equipo)` que nos permitirá identificar cada instante en que un equipo tomó el balón durante el último partido Argentina-Venezuela de la Copa América.



POSESIONES_ARGVEN

timestamp	equipo
16:21:18.418	Argentina
16:21:22.637	Argentina
16:21:29.827	NULL
16:21:42.402	Venezuela
16:21:46.535	Argentina
16:21:49.388	Argentina
16:21:55.109	NULL
...	...

Se pide:

a) Suponiendo que dispone de 100 bloques de memoria, complete el árbol de consulta de manera de definir un plan de ejecución, indicando el método de acceso utilizado para cada tabla y el método de junta con que se calculará la junta externa entre ambas. Señale las operaciones en que se hará *pipelining* y aquellas en que deberá materializar algún resultado.

b) Estime el costo del plan de ejecución diseñado. No considere el costo de almacenamiento del resultado final en disco.

c) Estime la cardinalidad del resultado en términos de cantidad de filas.

d) *(Bonus track)* Suponga ahora que la tabla `Posesiones_ArgVen(timestamp, equipo)` ya se encuentra en disco. La siguiente consulta SQL encuentra los pases de balón dentro de un mismo equipo y los registra como lapsos de tiempo en que el balón se encuentra en posesión de dicho equipo. Para ello, primero ordena la tabla por timestamp y luego utiliza la función `LAG` para obtener datos de la fila anterior a cada fila dentro de ese orden, calculando la diferencia de tiempo entre una y otra toma de posesión (es decir, el pase). Finalmente, se queda únicamente con los pases dentro de cada equipo.

```
WITH temp AS
  (SELECT equipo,
          timestamp - LAG(timestamp)
                      OVER (ORDER BY timestamp) as tiempo_posesion,
          LAG(equipo)
          OVER (ORDER BY timestamp) as equipo_anterior
   FROM Posesiones_ArgVen)
SELECT equipo, tiempo_posesion
FROM temp
WHERE equipo = equipo_anterior;
```

¿Cuál cree a grandes rasgos que será el costo de realizar una operación como esta si la tabla ya se encuentra ordenada en disco por timestamp? ¿Y si no estuviera ordenada? Justifique sus respuestas.

Con esta nueva tabla ya estaríamos en condiciones de calcular el tiempo de posesión total de cada equipo, agrupando por equipo y sumando los distintos tiempos de posesión.

equipo	tiempo_posesion
Argentina	00:00:04.219
Argentina	00:00:02.853
Venezuela	00:00:01.415
Argentina	00:00:08.380
...	...

3. *(Diseño relacional)* Dada la relación $R(A, B, C, D)$ y la dependencia multivaluada $B \twoheadrightarrow C$, muestre una instancia de la relación R que viole explícitamente dicha dependencia multivaluada, y una instancia que la respete.

   Si dicha dependencia multivaluada $B \twoheadrightarrow C$ pertenece al conjunto de dependencias de R, ¿es posible que B sea clave candidata? Justifique su respuesta.

4. *(Concurrencia y transacciones)* Mencione una forma de garantizar la recuperabilidad durante la ejecución concurrente de transacciones.

5. *(Seguridad)* El modelo de seguridad RBAC en *PostgreSQL* dispone de un privilegio de nombre `REFERENCES`, que permite a un usuario crear una tabla que referencia a otra de la cual no es dueño, a través de una clave foránea. Considere como ejemplo una tabla `Superhéroes(cod_héroe, nombre_héroe, nombre_vida_real, país_origen)` cuyo dueño es el usuario `Martín`. Para que el usuario `Pablo` pueda crear una tabla `Apariciones(nombre_historieta, número_ejemplar, cod_héroe)`, en donde el atributo `cod_héroe` referencia a la tabla `Superhéroes` de Martín, es necesario que previamente Pablo reciba permiso de `REFERENCES` sobre esa tabla, a través de un comando como el siguiente:

   ```
   GRANT REFERENCES ON TABLE Superhéroes TO Pablo;
   ```

   Explique de qué forma se podría violar la *confidencialidad* de la información de Martín si fuera posible referenciar libremente su tabla (es decir, sin recibir previamente dicho *GRANT*). *Ayuda: Piense en las restricciones que una clave foránea implica en el modelo relacional.*

   Nota: Suponga que el dueño del esquema es el superusuario `postgres`, y que tanto Pablo como Martín tienen permiso de `USAGE` y de `CREATE` sobre dicho esquema.

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

   a) Un atributo estático en Cassandra tomará un único y mismo valor en todas las *wide-rows* de la familia de columnas.

   b) En Cassandra la noción de agregado está representada por la *wide-row*.

   c) En los *log-structured merge trees* como los utilizados por Cassandra y MongoDB el árbol en memoria C_0 (*memtable*) suele implementarse con un B-tree.

   d) Garantizar consistencia eventual en una base de datos distribuida es más costoso que garantizar consistencia secuencial.

   e) El esquema de fragmentación implementado en MongoDB se corresponde con la *fragmentación vertical*.

   f) DynamoDB prioriza la disponibilidad (*availability*) por sobre la consistencia.