Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 13 de julio de 2022

TEMA 20221C1						**Padrón:** _____
Proc.		**AR**		**NoSQL**		**Apellido:** _____
CyT		**Snap.**		**Rec.**		**Nombre:** _____
						Cantidad de hojas: _____
Nota:						☐ 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. *(Procesamiento de Consultas)* La empresa de servicios de Internet y televisión *DadaNet* quiere determinar quiénes de sus clientes tienen facturas impagas en el barrio de Barracas. Para ello dispone de las siguientes dos tablas:

   - `Clientes(id_cliente, nombre, apellido, dirección, barrio, fecha_contratación)`
   - `Facturas(cod_factura, id_cliente, fecha, monto, estado)`

   La consulta SQL escrita para extraer está información fue:

   ```
   SELECT *
   FROM Clientes INNER JOIN Facturas USING(id_cliente)
   WHERE barrio = 'Barracas'
   AND estado = 'PENDIENTE';
   ```

   Se pide:

   a) Proponga un plan de ejecución adecuado, indicando de qué forma se accederá a cada tabla, y cómo se realizará la junta. Indique el costo de acceso para dicho plan, en términos de cantidad de bloques.

   b) Estime la cardinalidad del resultado, en términos de cantidad de tuplas.

   Asuma que dispone de `M=500` bloques de memoria disponibles, y considere que cuenta con los siguientes índices para las tablas:

   - IDX1 secundario en Clientes por (id_cliente), altura = 4
   - IDX2 de clustering en Clientes por (barrio), altura = 2

- IDX3 secundario en Facturas por (cod_factura), altura = 4
- IDX4 de clustering Facturas por (id_cliente), altura = 3
- IDX5 secundario en Facturas por (estado), altura = 2

Se dispone además de la siguiente información de catálogo respectiva a estas dos tablas:

CLIENTES	**FACTURAS**
n(Clientes) = 100.000	n(Facturas) = 500.000
B(Clientes) = 10.000	B(Facturas) = 50.000
V(barrio, Clientes) = 40	V(id_cliente, Facturas) = 100.000
	V(estado, Facturas) = 4

2. *(Álgebra relacional y SQL)* El *Ministerio de Salud* cuenta con las siguientes tablas que almacenan información sobre las dosis de vacunas aplicadas contra el COVID-19 en la Argentina:

   - `Personas(DNI, apellido, nombre)`
   - `Vacunaciones(DNI, fecha, lugar_vacunación, nro_lote)`
   - `Lotes(nro_lote, nombre_vacuna, empresa)`

   Un analista de datos quiere obtener cierta información de la base de datos, y envía su consulta a los operadores de la misma escrita en el lenguaje del Álgebra Relacional:

$$(\pi_{DNI,apellido,nombre,nombre\_vacuna}(Personas * Vacunaciones * Lotes)) \div \pi_{nombre\_vacuna}(\sigma_{nombre\_vacuna='AstraZeneca' \vee nombre\_vacuna='Sputnik'} Lotes)$$

   Se pide:

   a) Exprese en lenguaje coloquial el resultado de esta consulta, asumiendo que existen lotes de vacunas 'AstraZeneca' y 'Sputnik' en la tabla de Lotes.

   b) Indique cuál/es de las siguientes consultas son algebraicamente equivalentes a la consulta anterior:

   1)

$$\pi_{DNI,apellido,nombre}(Personas * Vacunaciones * (\sigma_{nombre\_vacuna='AstraZeneca'} Lotes)) \cup \pi_{DNI,apellido,nombre}(Personas * Vacunaciones * (\sigma_{nombre\_vacuna='Sputnik'} Lotes))$$

   2)

$$\pi_{nombre\_vacuna}(Personas * Vacunaciones) - (\sigma_{nombre\_vacuna='AstraZeneca'} Lotes) - (\sigma_{nombre\_vacuna='Sputnik'} Lotes)$$

   3)

$$\pi_{nombre\_vacuna}(\sigma_{nombre\_vacuna='AstraZeneca' \vee nombre\_vacuna='Sputnik'} Lotes) \div \pi_{DNI,apellido,nombre,nombre\_vacuna}(Personas * Vacunaciones)$$

4)

$$\pi_{DNI,apellido,nombre}(Personas * Vacunaciones * (\sigma_{nombre\_vacuna='AstraZeneca'} Lotes)) \cap \pi_{DNI,apellido,nombre}(Personas * Vacunaciones * (\sigma_{nombre\_vacuna='Sputnik'} Lotes))$$

5)

$$\pi_{DNI,apellido,nombre}(Personas) - \pi_{DNI,apellido,nombre}(Personas * Vacunaciones * (\sigma_{nombre\_vacuna='AstraZeneca'} Lotes)) - \pi_{DNI,apellido,nombre}(Personas * Vacunaciones * (\sigma_{nombre\_vacuna='Sputnik'} Lotes))$$

c) Traduzca la consulta al lenguaje SQL.

3. *(NoSQL)* A medida que avanzó la pandemia de COVID-19, el *Ministerio de Salud* continuó recolectando datos y los ingenieros de datos decidieron hacer una prueba piloto almacenando los datos sobre las vacunaciones en una base *MongoDB* distribuida. Para ello, crearon una colección llamada `vacunaciones` en que cada documento representa la aplicación de una dosis a una persona, como se puede ver en el siguiente ejemplo:

```
{ "DNI": "43914908",
  "apellido": Aviatky,
  "nombre": Enrique,
  "nro_dosis": 3,
  "fecha": ISODate("2022-03-08 15:14:49"),
  "posta_vacunación": "Centro Social N32 Esteban Ocampo"
  "localidad_vacunación": "Chacabuco"
  "lote": "XR30514B50A",
  "vacuna": "Moderna"
}
```

Esta base de datos se utilizará para resolver dos consultas puntuales:

- Dada una persona, encontrar su historial de vacunación (es decir, qué dosis recibió, qué vacunas se aplicó en cada dosis, y en qué fechas).

- Dada una localidad, calcular la cantidad de aplicaciones de cada vacuna efectuadas dentro de cada dosis. Para cada número de orden de dosis, se deberá generar un documento como el siguiente:

```
{ "nro_dosis": 1,
  "vacunas_aplicadas": [
      {
        "vacuna": "Astrazeneca",
        "cant_aplicaciones": 48202
      },
      {
        "vacuna": "Moderna",
        "cant_aplicaciones": 17580
      },
      ...
  ]
}
```

Se pide:

a) Sugiera una estrategia de *sharding* y de indexado que permita que estas consultas se respondan rápidamente (accediendo de la forma más directa que encuentre a los datos, y aprovechando la capacidad de procesamiento distribuido).

b) Escriba una consulta en *MongoDB* que resuelva la segunda consulta de las anteriormente mencionadas para la localidad de Balcarce, ordenando los resultados por el número de dosis (comenzando por la primera dosis).

4. *(Concurrencia y transacciones)* Dado el siguiente solapamiento de transacciones:

$$\mathtt{b}_{\mathtt{T}_1}; \mathtt{b}_{\mathtt{T}_2}; \mathtt{b}_{\mathtt{T}_3}; \mathtt{R}_{\mathtt{T}_1}(\mathtt{X}); \mathtt{R}_{\mathtt{T}_2}(\mathtt{X}); \mathtt{W}_{\mathtt{T}_2}(\mathtt{Y}); \mathtt{W}_{\mathtt{T}_1}(\mathtt{X}); \mathtt{c}_{\mathtt{T}_1}; \mathtt{R}_{\mathtt{T}_3}(\mathtt{Y}); \mathtt{W}_{\mathtt{T}_3}(\mathtt{Z}); \mathtt{c}_{\mathtt{T}_3}; \mathtt{R}_{\mathtt{T}_2}(\mathtt{Z}); \mathtt{c}_{\mathtt{T}_2}$$

Se pide:

a) Dibuje el grafo de precedencias del solapamiento.

b) Indique si el solapamiento es serializable. Justifique su respuesta.

c) Indique si el solapamiento es recuperable. Justifique su respuesta.

5. *(Snapshot Isolation)* Para el solapamiento del Ejercicio 4, analice qué cambiaría si la ejecución se realizara bajo *Snapshot Isolation*. Expliqué qué ocurriría con respecto a la serializabilidad y a la recuperabilidad, y si alguna transacción sería abortada.

6. *(Recuperación)* Indique si las siguientes afirmaciones sobre algoritmos de recuperación son verdaderas (V) o falsas (F), justificando su respuesta.

a) En el algoritmo REDO, cuando una transacción modifica un ítem `X` en memoria, el nuevo valor no debe enviarse a disco hasta después de que esa transacción haya hecho su *commit*.

b) El algoritmo UNDO/REDO tiende a generar archivos de *log* más grandes que el algoritmo UNDO.

c) En el algoritmo UNDO, cuando una transacción modifica un ítem `X` en memoria, el nuevo valor debe guardarse en el archivo de log, que debe volcarse a disco antes de que esa transacción haga su *commit*.

d) Si durante una recuperación bajo UNDO se encuentra un `END CKPT`, entonces se puede garantizar que no habrá que retroceder más allá del `BEGIN CKPT` asociado.