RESOLUCION SEGUNDO PARCIAL

1. *(Procesamiento de consultas)* Como recordará de exámenes previos, los siguientes esquemas de relación almacenan información sobre las multas de tránsito de la Ciudad de Buenos Aires:

   - `PERSONA(DNI, nombre, apellido, direccion, ciudad, celular)`
     // (21.454.201, 'Ramón', 'Mercury', 'Av. Rivadavia 500', 'Rosario', 5240-6544)
   - `VEHICULO(matricula, marca, modelo, fecha_VTV)`
     // ('AR 251 GH', 'RENAULT', 'DUSTER', '2022-12-05')
   - `MULTA(nro_multa, DNI_infractor, matricula, tipo, hora, fecha, lugar, importe)`
     // (1809, 21.454.201, 'AR 251 GH', 2, 12:23:21, '2022-01-01', 'RUTA 205 KM 34.5', 350.000)
   - `PROPIETARIO(DNI,matricula)`
     // (21.454.201, 'AR 251 GH')

   Tenga en cuenta que el infractor que cometió una multa con un vehículo no es necesariamente la misma persona registrada como propietaria de ese vehículo.

   Esta base de datos registra distintos tipos de multa, que van desde el tipo 1 (menos severa) al tipo 4 (muy severa). Entre las multas de tipo 4 se incluyen infracciones como "ingresar al Paseo del Bajo en contramano" o "cruzar semáforo en rojo sonando una sirena falsa", que determinan el quite de la licencia de por vida al propietario del vehículo.

   Por lo tanto, cuando una persona se presenta para renovar su licencia de conducir, además de chequear que la persona no haya cometido ella misma infracciones, se chequea que jamás se haya cometido una infracción de tipo 4 con ninguno de los vehículos que son de su propiedad. De lo contrario, la licencia es denegada. La siguiente consulta, en particular, es realizada

cuando Marlon Siniestra (DNI 18.324.715) se presenta a renovar su licencia en las oficinas de la Dirección de Tránsito:

```sql
SELECT *
FROM Propietario p INNER JOIN Multa m USING(matricula)
WHERE p.DNI = 18324715 AND m.tipo = 4;
```

Se pide:

a) Sugiera dos índices que puedan utilizarse para ejecutar más eficientemente esta consulta. Luego dibuje un plan de ejecución que haga uso de estos dos índices para la consulta. Específicamente, dibuje el árbol del plan de consulta y anote sobre el mismo los métodos de acceso o algoritmos que se utilizarán en cada paso.

b) Estime el costo del plan de ejecución que armó en el punto anterior, en términos de cantidad de accesos a bloques de disco.

Para el ejercicio considere que los índices son de tipo árbol y tienen altura 4. Además, considere para sus cálculos la siguiente información de catálogo:

PROPIETARIO	**MULTA**	**VEHICULO**
n(Propietario) = 600.000	n(Multa) = 300.000	n(Vehiculo) = 600.000
B(Propietario) = 100.000	B(Multa) = 30.000	B(Vehículo) = 200.000
V(DNI, Propietario) = 300.000	V(matricula, Multa) = 100.000	
	V(tipo, Multa) = 4	

a. Temenos un join y dos selecciones y nos piden dos índices. Seria ideal usar un índice en uno de los joins y el otro índice o en el otro join o en alguna de las selecciones.

No resultaría ideal poner los índices en ambas selecciones, porque si en el join no tengo acceso a índice, y tengo muchos datos, el join puede ser muy costoso.

Colocar un índice en la selección de tipo de multa tampoco resultaría muy conveniente, porque solo tengo 4 tipos de multa, la variabilidad es muy baja

En general me va a convenir hacer

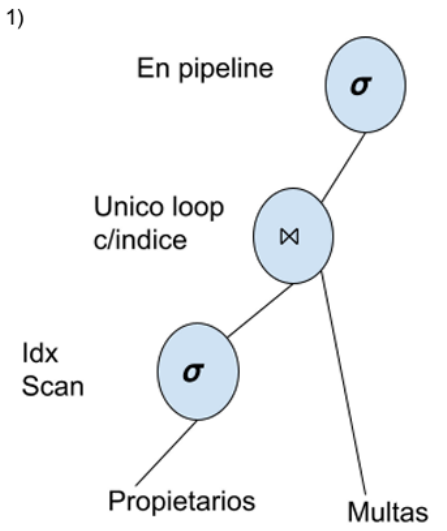
1. Una selección sobre la tabla, la mas restrictiva primero
2. El join
3. La ultima seleccion

Y usar los índices en la primera selección para quedarme con menos registros y en el join.

Observando los metadatos: Propietarios tiene 600.000 registros y 300.000 DNIs, por lo tanto en promedio hay 2 vehiculos por DNI. Si yo hago una selección por DNI puede obtener en promedio dos registros. Multas en cambio hay 300.000 multas y 4 tipos, y estimando ese 300.000/4 es mucho mayor que solo 2.

Hacemos esta selección, luego el join y por ultimo la selección de tipo de multa en pipeline, que eso no tiene costo agregado.

1)



Indice 1: En propietarios por DNI de clustering
Indice 2: En multas por matricula de clustering

Los hago de clustering para que mis índices funcionen mejor.

b. Para estimar el costo:

1ro se calcula el costo de la selección de propietarios, con un índice de clustering:

$$cost(S_5) = Height(I(A_i, R)) + \left\lceil \frac{n(R)}{V(A_i, R) \cdot F(R)} \right\rceil$$

Cost(Sel_1) = 4 + (B(Prop)/V(DNI, Prop) = 4 + 100.000/300.000 = 4.333 aproximadamente 5.

Luego, para la junta, debo estimar el costo teniendo en cuenta que ya no tengo la misma cantidad de atributos que antes.

Para la tabla propietarios, luego de aplicar la selección:

- El F se mantiene
- El n lo podemos estimar con la variabilidad: hacemos 600.000/300.000 = $\mathbf{n_{selec} = 2}$
- El B lo podemos estimar con la variabilidad: hacemos 100.000/300.000 = 1/3 redondeamos a $\mathbf{B_{selec} = 1}$ bloque

Con estos nuevos datos estamos en condiciones de calcular el costo del join, utilizando un método de loops anidados, en este caso por un índice de clustering, cuya formula es:

$$cost(R * S) = B(S) + n(S) \cdot (Height(I(A,R)) + \left\lceil \frac{n(R)}{V(A,R) \,.\, F(R)} \right\rceil)$$

A esto le tengo que restar la lectura de la tabla de Propietarios B(S) = B(Propietarios) porque ya la tengo en memoria con el B(Propietarios) original.

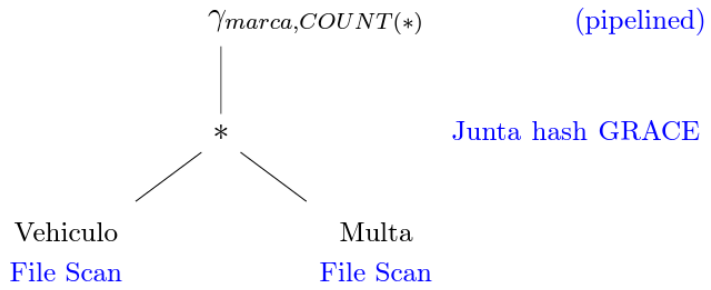
Teniendo el índice sobre la tabla de multas:

Cost(Multas * Propietarios) = 1 + 2 * (4 + 30.000/100.000) = 1 + 2 * (4 + 1) **– B(Prop)** = 1 + 10 – 1 = 10

Luego, la última selección tiene costo 0 porque tengo los datos levantados en memoria y no tengo que ir a buscarlos a disco

COSTO TOTAL = 5 + 10 = 15.

2. *(Procesamiento de consultas)* Para las mismas tablas del ejercicio anterior y la misma información de catálogo, se quiere calcular cuántas multas se han cometido con cada marca de vehículo. Asumiendo que no se cuenta con ningún tipo de índice y que se dispone de `M=20.000` bloques de memoria, se arma el siguiente plan de ejecución en el que la junta se realiza por el método de hash GRACE:



Se pide:

a) 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 térmidos de cantidad de bloques– tendrían las particiones en ese caso.

b) Indique cuál es el/los atributo/s a los que habrá que aplicar la función de hash en cada tabla, a efecto de construir las particiones.

c) Estime cuál sería el costo de realizar la junta planteada, en términos de cantidad de accesos a bloques de disco. Considere que el agrupamiento posterior se realiza en pipeline utilizando un diccionario en memoria, y por lo tanto no incurre en costos de acceso a disco.

a. Teniendo en cuenta que la proyección se hace en un pipeline, podemos asumir que consume muy poca memoria y por ende podemos usar el M para hacer la junta.

Yo voy a poder usar 20.000 bloques de memoria. De por si, no voy a poder tener mas de M – 1 particiones.

Decido que voy a usar 19.000 bloques de memoria para hacer el hasheo. Entonces hago 19.000 distintas particiones y cada bloque lo voy mandando a una partición distinta. Luego, me van a quedar particiones de una N cantidad de bloques

Para ver si esto es posible, las particiones tienen que ser tales que la **tabla mas chica tenga una cantidad de particiones que entre completamente en memoria.**

La tabla de vehículos tiene 200.000 bloques, y si tenemos 19.000 particiones, vamos a tener 200.000/19.000 = 11 bloques por partición aproximadamente de Vehiculos.

Veamos la tabla de Multas: tiene 30.000 bloques y con 19.000 particiones, me quedarías 30.000/19.000 = 2 bloques por partición aproximadamente. **Esta es la tabla mas chica.** Voy a tener más o menos 2 bloques por partición, de Multa.

En la 3ra etapa, yo levanto completamente todos los bloques de Multas de cada partición (osea dos bloques por partición), que es la tabla mas chica, y uno a uno levanto los bloques de vehículos, que son 11 por partición.

Por ende, voy a estar levantando 13 bloques, cuando tengo 20.000 bloques en memoria disponibles.

b. Para la tabla de vehículos, deberíamos hashear por el atributo de junta, es decir matricula, y para la otra tabla debe ser el mismo, así que la ta es *matricula* para ambos

c. Es decir que la proyección me da un costo 0, el cosot esta solo en la junta.

   Como es ***hash Grace,*** el costo es simplemente 3(B(Vehículo) + B(Multas) = 3 * (200.000 + 30.000) = 690.000

3. *(NoSQL)*

   a) *(MongoDB)* El *Club de Cinéfilos* quiere armar un ranking de actores que permita a sus miembros saber quiénes son los 100 actores a los que más vale la pena seguir. Para ello, cuentan con una base de datos de películas en MongoDB, que indica el puntaje en IMDB de cada película junto con el listado de actores de la película, tal como ejemplifica el siguiente documento:

```
{
  "_id": 10910355903998401931,
  "nombre_pelicula": "Interstellar",
  "genero_principal": "Ciencia Ficción",
  "puntaje_IMDB": 8.7,
  "actores": ['Matthew McConaughey', 'Jessica Chastain', 'Anne Hathaway',
     'Mackenzie Foy', 'Timothée Chalamet', 'Matt Damon', 'Michael Caine']
}
```

   Mientras discutían qué métrica utilizar para rankear a los actores, algunos sugerían usar el puntaje promedio en IMDB de sus películas como un valor representativo. Otros en cambio consideraban que se debía tomar el mejor puntaje de entre todas las películas en las que el actor participó, ya que si un actor participó en una película muy buena, entonces valía la pena seguirlo aún cuando su promedio fuera malo.

   Finalmente, se decidió por una estrategia híbrida en que se ordenará a los actores por su puntaje promedio en todas sus películas, pero se exluirá luego a aquellos actores cuya mejor película tenga un puntaje mayor o igual a 8.0.

   1) Escriba una consulta en MongoDB que devuelva el listado de los 100 mejores actores ordenados por este criterio, indicando para cada actor su nombre y apellido, su puntaje promedio, y la máxima puntuación obtenida por sus películas.
   2) Explique si la consulta anterior puede ser ejecutada con la colección shardeada por el atributo _id. En caso afirmativo, explique brevemente cómo podría realizarse el cálculo anterior en forma distribuida entre los shards y los servidores de agregación. En caso negativo, explique cuál debería ser el/los atributo/s de sharding, y cómo se realizaría el cálculo en forma distribuída en ese caso.

ERROR DE ENUNCIADO: vamos a excluir a los actores que tienen mejor película con mejor puntaje menor a 8.0

1. Por cada película voy a tener que hacer un unwind de los actores para quedarme con el score que tienen en cada película en la que estuvieron

Luego, voy a agrupar por actor, usando como _id del group a actores, y calcular el puntaje promedio del todas sus pelis y además quedarme con su máximo puntaje.

Cuando tenga esos datos, voy a excluir a los actores que tienen como puntaje máximo un puntaje menor a 8.0.

Ahora, puedo ordenar por el puntaje promedio, y limitar el output a los 100 mejores. Finalmente, proyecto los datos con los que me quiero quedar, que son el nombre del actor, el puntake

promedio y la máxima puntuación. Acá no hace falta hacer el $proyect porque ya tengo solo estos datos de haber hecho el agrupamiento

```
db.collection.aggregate([
  {
    $unwind:{
      path: "$actores"
      preserveNullAndEmptyArrays: false
    }
  },
  {
    $group: {
      _id: "$actores",
      "mejor": {
        $max: "$puntaje_IMDB",
      },
      "promedio" {
        $avg: "$puntaje_IMDB"
      }
    }
  },
  {
    $match: {
      "mejor": {
        $gte: 8
      }
    }
  },
  {
    $sort: {
      "promedio" : -1
    }
  },
  {
    $limit: {
      100
    }
  }
])
```

2. Si, puede ser shardeada por id. Se van a distribuir por ese numero _id, que es medio random, así que la distribución va a ser medianamente equitativa.

Como se resolvería: no todos los nodos van a tener todo el contexto de todas las películas del actor. En un nodo me puede haber quedado en una película de Matthew McConaghey y en otro otra. Para la máxima nota no hay problema, pero para el promedio no es tan sencillo, porque me va a estar calculando el promedio de los promedios. Calculando el 2do pormeido en el nodo de agregación cuando rejunto todos los valores de Matthew que me vinieron del resto de los nodos.

Cada nodo devolverá para cada actor, el máximo, la cantidad de peliculas y la suma de puntajes El nodo de agregación se quedará con el mejor de los máximos y para calcular el promedio hara suma total de puntajes / suma de cantidad de películas.

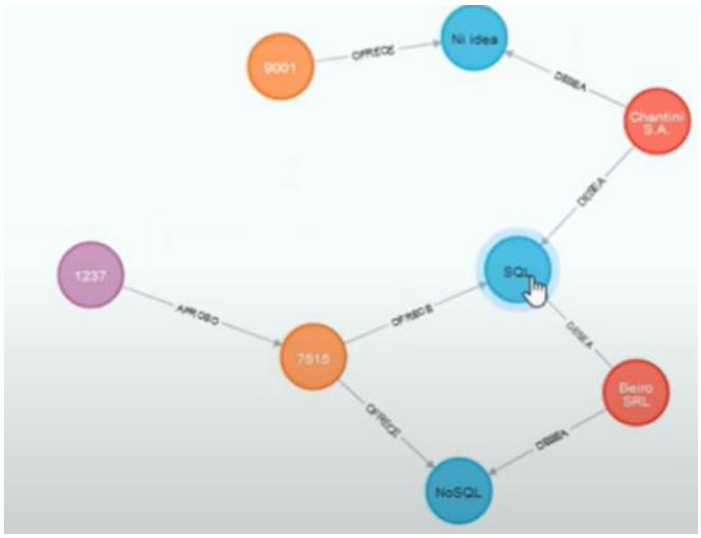
b) *(Neo4j)* La *Facultad de Ingeniería* quiere implementar un sistema de búsqueda de empleo en el que los estudiantes puedan cargar las asignaturas que cursaron y las empresas puedan cargar los conocimientos que requieren en cada búsqueda. El objetivo del sistema es recomendar a los estudiantes ofertas de empleo para las cuales estén potencialmente calificados, a partir de la información que se almacenará en una base de datos en Neo4j con los siguientes nodos y aristas:

```
(est:Estudiante {nombre: 'Guillermina', apellido: 'Fabbri',
                 username: 'laguille99', padrón: 109329})
(asign:Asignatura {nombre: 'Base de Datos', codigo: '7515'})
(conoc:Conocimiento {nombre: 'BASES DE DATOS NOSQL'})
(conoc2:Conocimiento {nombre: 'POSTGRESQL'})
(busq:Busqueda {nombre_empresa: 'El Sauce S.R.L.',
                fecha: '2023-10-27', contacto: 'rrhh@elsauce.com.ar'})
...
(est)-[:APROBO]->(asign)
(asign)-[:OFRECE]->(conoc)
(busq)-[:DESEA]->(conoc)
```

Así, cuando un estudiante se encuentre buscando empleo, el sistema le rankeará las búsquedas disponibles mostrando primero aquellas para las cuales el estudiante se encuentra mejor preparado, en el sentido de que cumpla con poseer la mayor cantidad de conocimientos que la misma exije.

Escriba una consulta en Neo4j que, dado un estudiante específico de padrón p liste las búsquedas ordenadas con dicho criterio, indicando los datos de la búsqueda y la cantidad de conocimientos deseados por la misma que el alumno posee.

Osea me van a venir ordenadas según cuantos conocimientos tiene cada alumno por búsqueda:



Acá el alumno tiene 2 conocimientos para la búsqueda de Beiro porque aprobó BDD, así que esta consulta viene primero.

1. Tenemos que ver, para un padron particular, alumnos que hayan aprobado materias y ver que conocimientos aporta eso. Osea ir de un nodo violeta a uno naranja a uno celeste a uno rojo
2. Agrupado por búsqueda, ver cuantos conocimientos deseados por búsqueda son ofrecidos por materias que aprobó el alumno.

```
MATCH
(e:Estudiante)-[:APROBO]-(a:Asignatura)-[:OFRECE]-(c:Conocimiento)-[:DESEA]-(b:Busqueda)
WHERE e.padron = 1237
WITH b, COUNT(DISTINCT c) AS cantidad
RETURN b, cantidad
ORDER BY cantidad DESC
```

En este caso es obvia la dirección, pero si no fuera obvia, por ejemplo un arco de persona es jefe de persona, tenemos que poner la flechita.

5. *(Concurrencia y Recuperación)*

   a) *(Concurrencia)* Dado el siguiente solapamiento de transacciones:

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

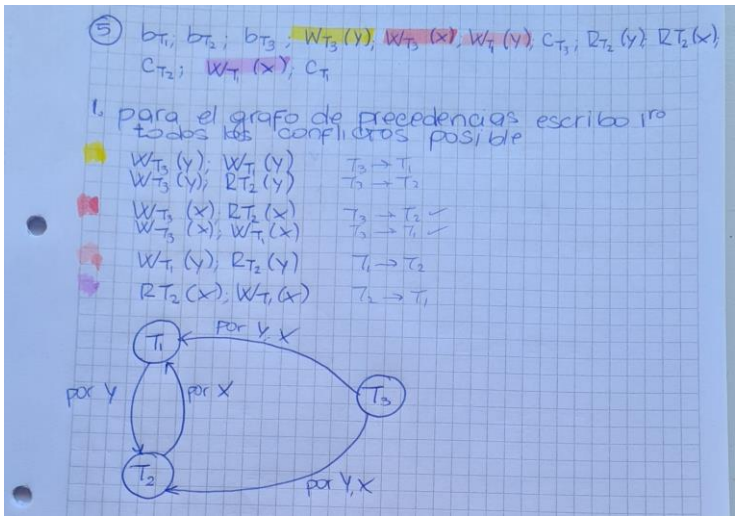
   1) Dibuje el grafo de precedencias del solapamiento.
   2) Indique si el solapamiento es serializable. Justifique su respuesta.
   3) Indique si el solapamiento es recuperable. Justifique su respuesta.

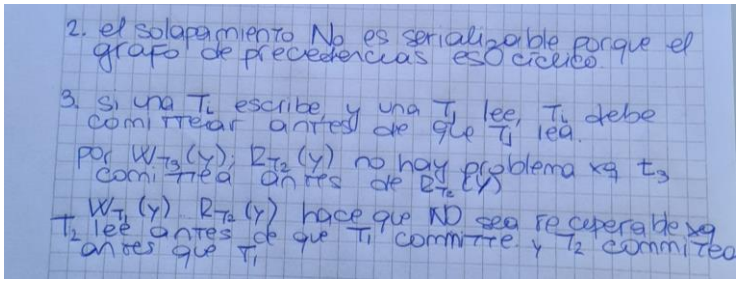
   b) *(Recuperación)* Un SGBD implementa el algoritmo de recuperación UNDO con checkpoint activo. Luego de una falla, el sistema encuentra el siguiente archivo de log:

```
01 (BEGIN, T1);                08 (WRITE T3, A, 8);
02 (BEGIN, T2);                09 (WRITE T2, B, 15);
03 (WRITE T1, A, 13);          10 (COMMIT, T2);
04 (WRITE T2, C, 7);           11 (END CKPT);
05 (COMMIT, T1);               12 (BEGIN, T4);
06 (BEGIN CKPT, T2);           13 (WRITE T3, B, 8);
07 (BEGIN, T3);                14 (WRITE T4, C, 9);
```

   Explique cómo se llevará a cabo el procedimiento de recuperación, indicando hasta qué punto del archivo de log se deberá retroceder, y qué cambios deberán ser realizados en disco y en el archivo de log.

a.





b. El primer checkpoint que se encuentra es un END CKPT, por lo que debemos retroceder hasta el 1er BEGIN CKPT. Voy a hacer un UNDO de las transacciones registradas como activas en ese begin checkpoint que NO comittearon.

Voy a tener que volver hasta la línea 2, donde se inicia la Transaccion activa mas antigua (T2).

A partir de ahí, voy a deshacer todas las transacciones que NO commitearon, que en este caso son la T3 y la T4. Reescribo:

- Para C: 9. Hago el flush de este dato a disco. ESCRIBO (ABORT, T4) en el log y hago flush del log a disco
- Para B: 8 y para A: 8. Hago el flush de este dato a disco. ESCRIBO (ABORT, T3) en el log y hago flush del log a disco