

Índice

1. Parcial Promocional 2023-12-06		3
1.1. Ejercicio 1(Procesamiento de consultas)	Índices arbol y costros	4
1.2. Ejercicio 2 (Procesamiento de consultas)	Hash GRACE	5
1.3. Ejercicio 3 (NoSQL)	MongoDB y colección Shardeada	5
1.4. Ejercicio 4 (CRT)	Grafo solapamiento, serializable?, recuperable?	6
1.5. Ejercicio 5 (Concurrencia y Transacciones)	UNDO ckpt activo	6
1.6. Ejercicio 6 (Recuperación)	UNDO ckpt activo	6
2. Parcial Promocional 2023-06-28		8
2.1. Ejercicio 1(Procesamiento de consultas)	Plan de ejecución y costos	9
2.2. Ejercicio 2 (Procesamiento de consultas)	Estimación de bloques y tuplas con histograma	9
2.3. Ejercicio 3 (NoSQL)	MongoDB y Neo4j	10
2.4. Ejercicio 4 (CRT)	Unrepeatable Read	10
2.5. Ejercicio 5 (Concurrencia y Transacciones)	UNDO/RÉDO ckpt activo	10
2.6. Ejercicio 6 (Recuperación)	UNDO/RÉDO ckpt activo	11
3. Coloquio 2023-12-13		12
3.1. Ejercicio 1 (SQL)	Plan de ejecución y estimar cardinalidad del resultado	13
3.2. Ejercicio 2 (Procesamiento de consultas)	Neo4j	14
3.3. Ejercicio 3 (NoSQL)	MongoDB	14
3.4. Ejercicio 4 (NoSQL)	Protocolo de Lock de 2 Fases	14
3.5. Ejercicio 5 (Concurrencia y Transacciones)	RÉDO ckpt activo	15
4. Coloquio 2023-08-02		16
4.1. Ejercicio 1 (SQL)	Costo con memoria limitada, cardinalidad de resultado	17
4.2. Ejercicio 2 (Procesamiento de consultas)	Neo4j	17
4.3. Ejercicio 3 (NoSQL)	Importancia de normalizar o no normalizar	18
4.4. Ejercicio 4 (NoSQL)	Protocolo de locks en dos fases	18
4.5. Ejercicio 5 (CRT)	Solapamiento, serializable?, recuperable?	19
5. Coloquio 2023-07-26		20
5.1. Ejercicio 1 (CRT)	Costo con memoria limitada, cardinalidad de resultado	21
5.2. Ejercicio 2 (Procesamiento de consultas)	Plan de ejecución y costos	21
5.3. Ejercicio 3 (NoSQL)	Familias de columnas Cassandra	22
5.4. Ejercicio 4 (NoSQL)	Hashing consistente en Dynamo	23
5.5. Ejercicio 5 (Concurrencia y Transacciones)	Protocolo de locks en dos fases	24
5.6. Ejercicio 6 (Recuperación)	V/F sobre REDO	25
6. Coloquio 2023-07-19		26
6.1. Ejercicio 1 (Procesamiento de Consultas)	Proporcionar indices y plan de ejecución	27
6.2. Ejercicio 2 (SQL)	MongoDB	28
6.3. Ejercicio 3 (NoSQL)	V/F	28
6.4. Ejercicio 4 (NoSQL)	V/F 2PL	29
6.5. Ejercicio 5 (Concurrencia y Transacciones)	Vent y desv de Snapshot Isolation	30
7. Coloquio 2023-07-12		31
7.1. Ejercicio 1 (CRT)	Estimar costo k para Hash GRACE, y atr de hasheado	33
7.2. Ejercicio 2 (Procesamiento de Consultas)	MongoDB	33
7.3. Ejercicio 3 (Diseño Relacional)	UNDO ckpt activo	35
7.4. Ejercicio 4 (NoSQL)	Solapamiento 2PL, serializable?, rollback?	35

8. Coloquio 2023-07-05		37
8.1. Ejercicio 1 (Modelado)	Estimar costo	38
8.2. Ejercicio 2 (Procesamiento de Consultas)		38
8.3. Ejercicio 3 (CRT)		38
8.4. Ejercicio 4 (SQL)	Neo4j	39
8.5. Ejercicio 5 (NoSQL)	Teoría agregado	39
8.6. Ejercicio 6 (NoSQL)		39
9. Coloquio 2023-03-01		40
9.1. Ejercicio 1 (CRT)	Plan de ejecución + costo	41
9.2. Ejercicio 2 (Procesamiento de Consultas)		41
9.3. Ejercicio 3 (SQL)	V/F Dynamo	42
9.4. Ejercicio 4 (NoSQL)	Neo4j	42
9.5. Ejercicio 5 (NoSQL)	V/F Conurrencia	42
9.6. Ejercicio 6 (Concurrencia y Transacciones)		43
10. Coloquio 2023-02-22		44
10.1. Ejercicio 1 (SQL)	Costo y tamaño de resultado	45
10.2. Ejercicio 2 (Procesamiento de Consultas)	V/F Mongo con sharding	45
10.3. Ejercicio 3 (NoSQL)	V/F LSM trees	46
10.4. Ejercicio 4 (NoSQL)	Concurrencia extraña con timestamps	46
10.5. Ejercicio 5 (Concurrencia y Transacciones)	UNDO ckpt activo	46
10.6. Ejercicio 6 (Recuperación)		47
11. Coloquio 2023-02-15		48
11.1. Ejercicio 1 (CRT)	Neo4j	49
11.2. Ejercicio 2 (NoSQL)	Costo y estimación de memoria	49
11.3. Ejercicio 3 (Procesamiento de Consultas)		49
11.4. Ejercicio 4 (SQL)	V/F sobre un solapamiento	50
11.5. Ejercicio 5 (Concurrencia y Transacciones)		50
12. Coloquio 2022-12-21		52
12.1. Ejercicio 1 (CRT)	Costos y que pasaría con x memoria	53
12.2. Ejercicio 2 (Procesamiento de Consultas)		53
12.3. Ejercicio 3 (SQL)	MongoDB	54
12.4. Ejercicio 4 (NoSQL)	V/F 2PL y Snapshot Isolation	54
12.5. Ejercicio 5 (Concurrencia y Transacciones)	UNDO/REDO ckpt activo	54
12.6. Ejercicio 6 (Recuperación)		54
13. Apéndice		56
13.1. (CRT) Caso de uso de V. Ejemplo de teórica		56
13.2. (NoSQL) Ayuda Cassandra - Diagrama Chebotko		56
13.3. (Concurrencia) Implicancias útiles		56
13.4. Costos		57

1. Parcial Promocional 2023-12-06

Dpto. de Computación (Fac. de Ingeniería - Universidad de Buenos Aires) TEMA 2023241 1

Base de Datos (75.15 / 75.28 / 95.05)

Segundo Parcial Promocional

TEMA 2023241	Proc.		Fecha: 6 de diciembre de 2023
	NoSQL		Padrón: _____
	CRT		Apellido: _____
	CyR		Nombre: _____
Corrigió:		Cantidad de hojas: _____	
Nota:		<input type="checkbox"/> Aprobado <input type="checkbox"/> Insuficiente	

Criterio de aprobación: El examen está compuesto por 7 ítems, cada uno de los cuales se corrige como B-/Reg-/M-. El examen se aprueba con nota mayor o igual a 4(cuatro) y la condición de aprobación es desarrollar un ítem bien (B-/B-) en al menos 3 de los 4 grupos de ejercicios (procesamiento de consultas, NoSQL, CRT, concurrencia/recuperación). Adicionalmente, no deberá haber más de dos ítems mal o no desarrollados.

- (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, dirección, 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(DNI_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 quiebre 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

Dpto. de Computación (Fac. de Ingeniería - Universidad de Buenos Aires) TEMA 2023241 3

- 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.

3. (NoSQL)

- (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:

```

1 {
2   "_id": 10910355903998401931,
3   "nombre_pelicula": "Interstellar",
4   "genero_principal": "Ciencia Ficción",
5   "puntaje_IMDB": 8.7,
6   "actores": [ 'Matthew McConaughey', 'Jessica Chastain', 'Anne Hathaway',
7   'Mackenzie Foy', 'Timothée Chalamet', 'Matt Damon', 'Michael Caine' ]
7 }

```

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 ordenaría a los actores por su puntaje promedio en todas sus películas, pero se excluiría luego a aquellos actores cuya mejor película tenga un puntaje mayor o igual a 8.0.

- 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.
- Explique si la consulta anterior puede ser ejecutada con la colección shardead 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 distribuida en ese caso.

- (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:

```

1   (est:Estudiante {nombre: 'Guillermina', apellido: 'Fabri',
2   username: 'laguille99', padrón: 1093291})
3   (asign:Asignatura {nombre: 'Base de Datos', codigo: '7515'})
4   (conoc:Conocimiento {nombre: 'BASES DE DATOS NOSQL'})
5   (conoc2:Conocimiento {nombre: 'POSTGRESQL'})
6   (busq:Busqueda {nombre_empresa: 'El Sauce S.R.L.', 

```

3. a.2) La consulta anterior no puede ser ejecutada eficientemente en una colección shardead por el atributo _id. Esto se debe a que el atributo _id es único para cada documento y no proporciona una distribución uniforme de los datos para la agregación que queremos realizar.

Atributo de sharding recomendado:

Un atributo de sharding más adecuado sería uno que asegure una distribución uniforme de los documentos entre los shards. En este caso, usar el campo actores como atributo de sharding puede ser más efectivo, dado que queremos realizar agregaciones por actores.

Dpto. de Computación (Fac. de Ingeniería - Universidad de Buenos Aires) TEMA 2023241 2

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

```

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

```

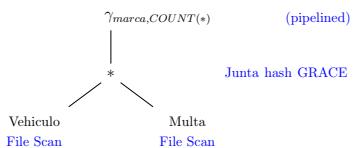
Se pide:

- 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.
- 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(Vehículo) = 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

- (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:

- 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 términos de cantidad de bloques– tendrían las particiones en ese caso.
- 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.

Dpto. de Computación (Fac. de Ingeniería - Universidad de Buenos Aires) TEMA 2023241 4

```

7   fecha: '2023-10-27', contacto: 'rrhh@elsauce.com.ar')
8   ...
9   (est)-[:APROBO]->(asign)
10  (asign)-[:OFRECE]->(conoc)
11  (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 exige.

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.

- (CRT) Considere las mismas relaciones del ejercicio 1, y escriba una expresión en *Cálculo Relacional de Tuplas* que encuentre el nombre, apellido y celular de aquellas personas propietarias de algún vehículo cuya última VTV (Verificación Técnica Vehicular) haya sido hace más de un año.

5. (Concurrencia y Recuperación)

- (Concurrencia) Dado el siguiente solapamiento de transacciones:

```

    bT1; bT2; bT3; WT3(Y); WT3(X); cT1; RT2(Y); RT2(X); cT2; WT1(X); cT1
1  fecha: '2023-10-27', contacto: 'rrhh@elsauce.com.ar')
2  ...
3  (est)-[:APROBO]->(asign)
4  (asign)-[:OFRECE]->(conoc)
5  (busq)-[:DESEA]->(conoc)

```

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.

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

```

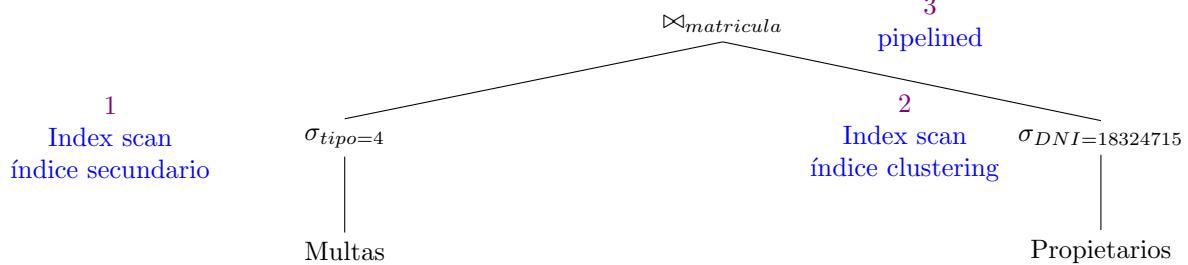
Explique cómo se llevará a cabo el procedimiento de recuperación, indicando hasta qué punto del archivo de log se deberá retroceder, y qué cambios deberán ser realizados en disco y en el archivo de log.

Padrón: _____ Apellido y nombre: _____

1.1. Ejercicio 1(Procesamiento de consultas)

Respuesta

Imaginemos dos planes de consulta, el primero siguiendo la regla de poner las selecciones lo más inmediatas posibles dentro del plan de consulta:



Suponiendo que no hay problemas de memoria, los costos serían:

tipo no es clave

- La selección del tipo de multa, denotada por 1, al ser una búsqueda por **índice secundario**, se tiene que

$$C_1 = H(I(tipo, Multas)) + \lceil \frac{n(Multas)}{V(tipo, Multas)} \rceil = 4 + \lceil \frac{300,000}{4} \rceil = 75,004$$

Y el resultado consta de $\frac{n(Multas)}{V(tipo, Multas)} = 75,000$ filas

es clave y arbol (?)

- Para calcular el costo de 2, se tiene un proceso similar al anterior, sólo que en este caso el índice es de clustering

$$C_2 = H(I(DNI, Proprietarios)) + \lceil \frac{B(Proprietarios)}{V(DNI, Proprietarios)} \rceil = 4 + \lceil \frac{100,000}{300,000} \rceil = 5$$

En este paso, la cantidad de tuplas resultante es $\frac{n(Proprietarios)}{V(DNI, Proprietarios)} = 2$

- El costo de la junta sería nulo, suponiendo que ambos resultados previos entran en memoria:

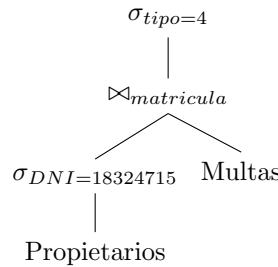
$$C_3 = 0$$

Luego, el costo total es:

$$C_T = C_1 + C_2 + C_3$$

$$C_T = 75,004 + 5 + 0 = 75,009$$

Para comparar costos, ahora planteamos un plan de consulta distinto



- La selección del DNI, al ser una búsqueda por índice de clustering, se tiene que

$$C'_1 = H(I(DNI, Proprietarios)) + \lceil \frac{B(Proprietarios)}{V(DNI, Proprietarios)} \rceil = 4 + \lceil \frac{100,000}{300,000} \rceil = 5$$

En este paso, la cantidad de tuplas resultante es $\frac{n(Proprietarios)}{V(DNI, Proprietarios)} = 2$

- Junta por único loop: suponiendo que también se tiene un índice de clustering en la tabla de Multas por **matricula** y que la primera etapa se encuentra en pipeline, el costo se puede expresar como:

$$C'_2 = n(1) * (H(I(matricula, Multas)) + \lceil \frac{B(Multas)}{V(matricula, Multas)} \rceil)$$

$$C'_2 = 2 * (4 + \lceil \frac{30,000}{100,000} \rceil)$$

$$C'_2 = 10$$

3. El costo de la última selección sería nulo, suponiendo que se encuentra en pipeline:

$$C'_3 = 0$$

Luego, el costo total es:

$$C_T = C'_1 + C'_2 + C'_3 = 5 + 10 + 0 = 15$$

1.2. Ejercicio 2 (Procesamiento de consultas)

a) Respuesta

Para definir la cantidad de particiones necesarias k para llevar a cabo la junta HASH GRACE es importante considerar las siguientes limitaciones:

$$V(PK, Tabla) = n(PK) \text{ (creo)}$$

- $k \leq V(\text{matricula}, \text{Multas}) = 300,000$, $k \leq V(\text{matricula}, \text{Vehiculo}) = 600,000$: no tendría sentido generar más particiones que la cantidad de valores distintos que puede tomar el atributo de junta, abarcarián la misma información, pero habría particiones vacías (si tienen tamaño suficiente).
- $k \leq M$: En la primera etapa, donde hay que leer R ó S (tablas a modo de ejemplo) y armar su particionado, hay que tomar al menos un bloque para armar cada partición R_i y sobre un bloque de memoria para hacer desfilar al otro grupo.
- $\frac{B(\text{Multas})}{k}, \frac{B(\text{Vehiculo})}{k} \leq M$: El tamaño de cada partición no debe superar la cantidad de memoria disponible para que en la segunda etapa se pueda realizar la junta de cada particiones (R_i, S_i) .

Dicho esto, se generan $k = \sqrt{\max(B(R), B(S))} = \sqrt{B(\text{Vehiculo})} = 447$. particiones de tamaño 448 bloques cada una. Se toma $\max(B(R), B(S))$ para asegurar que todas las tuplas de la tabla más grande que tengan el mismo valor de atributo de junta entren en una única partición.

b) Respuesta

El atributo por el que se debe hashear cada tabla **SIEMPRE** es el atributo de junta.

c) Respuesta

El costo de la junta es

$$C = 3(B(\text{Vehiculo}) + B(\text{Multas})) = 3(230,000) = 690,000 \quad (1)$$

1.3. Ejercicio 3 (NoSQL)

Respuesta

```
a) {
    $unwind: "$actores"      desarrolla el array de actores dando una entrada por pelicula por actor
},,
{
    $group:{                (arriba mas info)
        _id:"$actores",   agrupa por actores
        nombre: { $first: { $split: ["$actores", " "] } }, me separa por nombre y apellido como un array
        puntaje_promedio: { $avg: "$puntaje_IMDB" }, promedio de IMDB
        max_puntaje: { $max: "$puntaje_IMDB" }, puntaje maximo de IMDB
        cantidad_peliculas: { $sum: 1 } // Contar la cantidad de peliculas por actor
                                         el 1 indica sumar de a 1
    },
    {$sort:{"puntaje_promedio": -1}}, ordena por puntaje promedio, 1 asce -1 desc
    {$project:{               se olvido de filtrar menor a 8.
        _id:0,                  { $match: { max_puntaje: { $lt: 8 } } },
        nombre:1,                 lt: less than, gt: greater than, gte: greater than or equal, lte: less than or equal
        puntaje_promedio:1,
        max_puntaje:1             proyecta la informacion que queremos. 0 excluye y 1 incluye
    },
    { $limit: 100} se olvido de limitar en 100
},
```

```
b) MATCH (e:Estudiante:{padron:p}),  
      (a:Asignatura),  
      (c:Conocimiento),  
      (b:Busqueda),  
      caminos= b-[:DESEA*]->c<-[:OFRECE*]-a<-[:APROBO*]-e  
RETURN b, COUNT(caminos)  
ORDER BY COUNT(caminos)
```

El asterisco (*) en una relación especifica un patrón de longitud variable, lo que permite buscar caminos de longitud variable en el grafo. Vamos a desglosar la consulta que has proporcionado para explicar detalladamente lo que hace cada parte, incluyendo el significado del asterisco.

3. a.2)
Ejecución distribuida:
En un entorno distribuido, cada shard realiza la agregación en paralelo para los actores que contiene. Los pasos son los siguientes:

Desenrollar el array de actores (\$unwind):
Cada shard desenrolla los documentos localmente.

Agrupar documentos por actor (\$group):
Cada shard agrupa los documentos localmente.

Filtrar documentos donde max_puntaje es menor a 8 (\$match):
Cada shard aplica el filtro localmente.

Ordenar documentos por puntaje_promedio (\$sort):
Cada shard ordena los documentos localmente.

Limitar el resultado a 100 actores (\$limit):
Cada shard limita los documentos localmente.

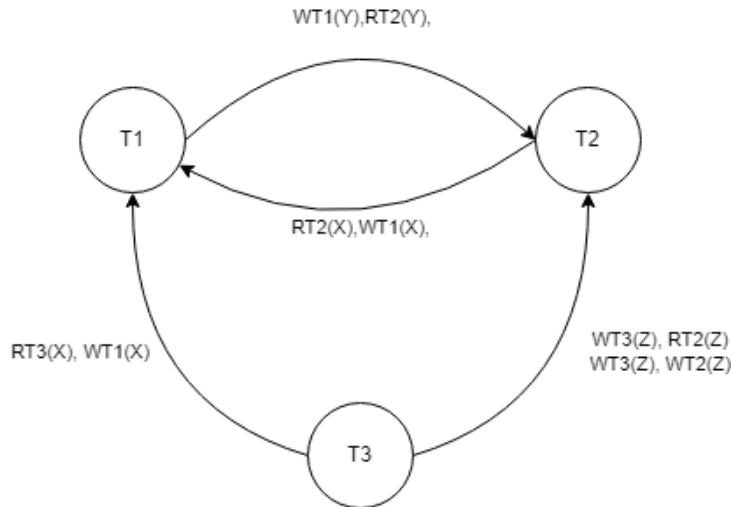
1.4. Ejercicio 4 (CRT)

Respuesta

Considere las mismas relaciones del ejercicio 1, y escriba una expresión en Cálculo Relacional de Tuplas que encuentre el nombre, apellido y celular de aquellas personas propietarias de algún vehículo cuya última VTV (Verificación Técnica Vehicular) haya sido hace más de un año

$$\{p.nombre, p.apellido, p.celular \mid \begin{aligned} &Person(p) \wedge (\exists pr)(Propietario(pr) \wedge \\ &p.DNI = pr.DNI \wedge (\exists v)(Vehiculo(v) \wedge \\ &v.matricula \leq pr.matricula \wedge \\ &v.fecha_vtv \leq' 2022 - 12 - 05'))\} \}$$

1.5. Ejercicio 5 (Concurrencia y Transacciones)



Como el grafo tiene ciclos, **no** es serializable.

Un solapamiento es recuperable si ninguna transacción T_i commitea hasta que las T_j que escriben elementos antes que T_i las leyera hayan commiteado \Rightarrow no es recuperable porque T_2 commitea antes que T_1 y T_1 escribe datos que T_2 lee.

2.6. Ejercicio 6 (Recuperación)

Respuesta

En este algoritmo, todos los datos modificados antes del BEGIN CKPT estarán en disco. Esto implica que:

- Para transacciones que **no** hayan comitteado:
hay que deshacer todas sus instrucciones. Se recorre el log de adelante hacia atrás aplicando cada uno de los WRITE para restaurar el valor anterior en disco.
Nota: Tomando como referencia BEGIN CKPT: las no commiteadas después ese punto, deben ser deshechas. También se tienen en cuenta las t_{act} no commiteadas. Las que commitearon antes de este punto ya tienen sus valores guardados en disco y sus instrucciones correspondientes en el log, en disco.
- Para transacciones que **sí** hayan comitteado:
no hay que rehacer las instrucciones previas al BEGIN CKPT, sólo las posteriores. Se recorre de atrás hacia adelante volviendo a aplicar cada uno de los WRITE de estas transacciones, para asegurar que quede asignado el nuevo valor de cada ítem.

Finalmente, por cada transacción de la que no se encontró el COMMIT se escribe (ABORT, T_i) en el log y se hace flush del log a disco.

Luego vemos que

1. T_3 commiteó antes del BEGIN CKPT, por lo que no habrá que hacer nada.
2. T_1 y T_2 commitearon después del BEGIN CKPT, por lo que habrá que rehacer sus instrucciones posteriores a este punto.
3. T_4 no commiteó, por lo que habrá que deshacer sus instrucciones.

Los pasos a seguir son:

1. Escribir C=2 por T_4 línea 17
2. Escribir E=8 por T_4 línea 12
3. Escribir D=10 por T_2 línea 11
4. Escribir B=3 por T_1 línea 14

Finalmente, se escribe (ABORT, T_4) en el log y se hace flush del log a disco.

2. Parcial Promocional 2023-06-28

Dpto. de Computación (Fac. de Ingeniería - Universidad de Buenos Aires) TEMA 2023141 1

Base de Datos (75.15 / 75.28 / 95.05)

Segundo Parcial Promocional

TEMA 2023141	Proc.		Fecha: 28 de junio de 2023
	NoSQL		Padrón: _____
	CRT		Apellido: _____
	CyR		Nombre: _____
Corrigió:		Cantidad de hojas: _____	
Nota:		<input type="checkbox"/> Aprobado <input type="checkbox"/> Insuficiente	

Criterio de aprobación: El examen está compuesto por 7 ítems, cada uno de los cuales se corrige como B-/B-/Reg-/M-. El examen se aprueba con nota mayor o igual a 4 (cuatro) y la condición de aprobación es desarrollar un ítem bien (B-/B-) en al menos 3 de los 4 grupos de ejercicios (procesamiento de consultas, NoSQL, CRT, concurrencia/recuperación). Adicionalmente, no deberá haber más de dos ítems mal o no desarrollados.

1. (*Procesamiento de consultas*) Gustavo trabaja en la Dirección de Tránsito del municipio de Tandil. Cada vez que un conductor se acerca para renovar su licencia de conducir, Gustavo debe verificar que el mismo no posea ninguna multa de tránsito pendiente de pago. Para consultar el estado de deuda del conductor, Gustavo accede a una base de datos relacional con las siguientes dos tablas:

```
■ licencias(nro_licencia, DNI_conductor, fecha_ultima_renovacion)
  // (740523, 41341989, 2019-03-02)

■ multas(patente_vehiculo, fecha, DNI_propietario, pagada)
  // ('AB415ZD', 2022-11-15, 41341989, False)
```

La base de datos también dispone de un índice secundario por **DNI_propietario** y de un índice de clustering por **pagada**, ambos sobre la tabla *Multas*.

Cuando Clemenciano Taqui llega para renovar su licencia (nro. 740523), Gustavo ejecuta la siguiente consulta en la base de datos:

```
SELECT *
FROM licencias INNER JOIN multas ON DNI_conductor=DNI_propietario
WHERE nro_licencia=740523
AND pagada IS FALSE;
```

Se pide:

- a) Proponga un plan de ejecución eficiente para esta consulta. Para ello dibuje un plan de consulta y anote sobre el mismo los métodos de acceso o algoritmos que se utilizarán en cada paso.

Dpto. de Computación (Fac. de Ingeniería - Universidad de Buenos Aires) TEMA 2023141 3

```
8      "id": 6253282,
9      "id_str": "6253282",
10     "name": "SMN Argentina",
11     "screen_name": "SMN_Argentina",
12     "followers_count": 173280,
13     "friends_count": 694
14   }
15   "entities":
16   {
17     "hashtags": ['#lluvias', '#Argentina'],
18     "user_mentions": [],
19     "media": []
20   },
21   "place":
22   {
23     "country": "Argentina",
24     "country_code": "AR"
25   },
26   "source": "Twitter Web Client"
27 }
```

Extraeremos los *trending topics* en base a los hashtags indicados en la publicación. Para ello se pide:

- Escriba una consulta en MongoDB que devuelva los 5 hashtags más frecuentemente utilizados en tweets escritos desde Argentina.
 - Sugiera una forma conveniente de shardear la colección a efectos de parallelizar esta consulta, e indique también si le convendría tener algún índice secundario en cada nodo de procesamiento para evitar tener que hacer un File Scan sobre todos los documentos almacenados en cada nodo. Justifique sus respuestas.
- b) (*Neo4j*) Una base de datos en Neo4j posee información sobre cada uno de los vinos que se producen en Argentina, indicando las bodegas que los producen y las cepas de uva con las que se produce cada vino, como se muestra en el siguiente ejemplo:

```
1   (v:Vino {nombre: '33 orientales'}, puntaje: 37.4, precio: 315.2)
2   (b:Bodega {nombre: 'El charrúa'})
3   (c1:Cepa {nombre: 'Cabernet Sauvignon'})
4   (c2:Cepa {nombre: 'Malbec'})
5   ...
6   (b)-[:FABRICA]-(v)
7   (v)-[:ELABORADO_CON]-(c1)
8   (v)-[:ELABORADO_CON]-(c2)
```

Los puntajes de los vinos oscilan entre 0 y 100. En particular, el vino del ejemplo anterior es fabricado por la bodega *El charrúa*, se elabora con cepas de cabernet sauvignon y malbec, y posee un puntaje de 37.4. Escriba una consulta en Neo4j que encuentre el nombre de la bodega que fabrique el vino de cepa Syrah monovarietal (es decir, que se elabora con una única cepa) de mayor puntaje en Argentina, mostrando el nombre de la bodega, el nombre del vino y su precio.

Nota: Le sugerimos utilizar la estructura `MATCH ... WHERE [NOT] EXISTS { MATCH pattern } ...` como parte de la solución.

4. (*CRT*) Considere las mismas relaciones del ejercicio 1, y escriba una expresión en *Cálculo Relacional de Tuplas* que encuentre los números de licencia de los conductores que tienen al menos dos multas impagadas.

Dpto. de Computación (Fac. de Ingeniería - Universidad de Buenos Aires) TEMA 2023141 2

- 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:

LICENCIAS	MULTAS
n(Licencias) = 30.000	n(Multas) = 60.000
B(Licencias) = 5.000	B(Multas) = 10.000
V(DNI_conductor, Licencias) = 30.000	V(DNI_propietario, Multas) = 15.000

2. El Instituto Nacional de Coordinación Agrícola se ocupa de brindar asistencia a los productores rurales, contando con un extenso equipo de asesores asignados a las distintas provincias del país. Su base de datos cuenta con los siguientes esquemas de relación con datos sobre los asesores y los productores rurales:

- Asesores(legajo, nombre, email, prov_asignada)
- Productores(guit, nombre, email, hectáreas, prov_origen)

Se quiere construir una tabla para poner en contacto a los asesores con los productores de las provincias que les fueron asignadas, a través de la siguiente operación de junta:

Asesores $\bowtie_{prov_asignada=prov_origen}$ *Productores*

Estime la cardinalidad del resultado de esta junta en términos de cantidad de tuplas y en términos de cantidad de bloques, utilizando el siguiente histograma de frecuencias extraído de la información de catálogo que muestra la frecuencia de las 5 provincias principales de cada tabla. Considere además que $V(prov_asignada, Asesores) = V(prov_origen, Productores) = 23$ y que el factor de bloqueo de ambas tablas es de 10.

	Santa Fe	Bs.As.	Córdoba	Río Negro	Entre Ríos	otras
Asesores.prov_asignada	90	110	70	60	45	280
Productores.prov_origen	1700	2200	1100	640	720	3100

3. (NoSQL)

- a) (*MongoDB*) Accedemos a una base de datos en MongoDB con tweets de mayo de 2023 de todo el mundo, a fin de estimar cuáles fueron los *trending topics* del mes en Argentina. Cada tweet de esta base de datos tiene una estructura similar a la siguiente:

```
1 {
2   "created_at": "Mon May 29 20:19:24 +0000 2023",
3   "id": 1050118621198921728,
4   "id_str": "1050118621198921728",
5   "text": "#Tras las #lluvias intensas de los últimos días en #Argentina, se observan cambios en la humedad del suelo y la vegetación del sur del Litoral y norte de Bs As. En colores oscuros también se evidencia el agua acumulada en lagunas y ríos.",
6   "user": {
7     }
```

Dpto. de Computación (Fac. de Ingeniería - Universidad de Buenos Aires) TEMA 2023141 4

5. (Concurrencia y Recuperación)

- a) (*Concurrencia*) Se dice que una transacción T_1 realiza una Lectura no Repetible ó *Unrepeatable Read* cuando la misma lee un ítem X , luego otra transacción T_2 escribe ese mismo ítem, y posteriormente T_1 vuelve a leer el ítem, encontrando un valor distinto al anteriormente leído. Indique si las siguientes afirmaciones sobre la Lectura no Repetible son verdaderas ó falsas. Justifique cada una de sus respuestas.

- En un solapamiento recuperable puede ocurrir una Lectura no Repetible.
- En un solapamiento que evita *rollbacks* en cascada puede ocurrir una Lectura no Repetible.
- Aplicando el Protocolo de Lock de Dos Fases (2PL) puede ocurrir una Lectura no Repetible.
- Bajo el nivel de aislamiento *Read Committed* definido en el estándar SQL puede ocurrir una Lectura no Repetible.

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

```
01 (BEGIN, T1);
02 (WRITE, T1, A, 5, 3);
03 (BEGIN, T2);
04 (WRITE, T2, B, 4, 8);
05 (WRITE, T1, C, 3, 2);
06 (BEGIN, T3);
07 (WRITE, T3, D, 15, 12);
08 (COMMIT, T3);
09 (BEGIN CKPT, {T1, T2});
10 (BEGIN, T4);
11 (WRITE, T2, D, 12, 10);
12 (WRITE, T4, E, 8, 18);
13 (COMMIT, T2);
14 (WRITE, T1, B, 8, 3);
15 (END CKPT);
16 (COMMIT, T1);
17 (WRITE, T4, C, 2, 1);
```

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

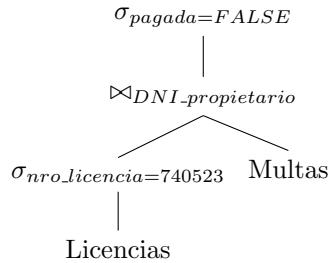
Padrón: _____

Apellido y nombre: _____

2.1. Ejercicio 1(Procesamiento de consultas)

Respuesta

Se plantea el siguiente plan de consulta



- La selección del número de licencia, al ser un *file scan*, se tiene que

$$C_1 = B(\text{Licencias}) = 5,000$$

el enunciado no dice que tenga índice, por lo que asumo que no

En este paso, la cantidad de tuplas resultante es $n_1 = 1$ que entra en $B_1 = 1$ bloque, pues se toma un único valor del atributo que es clave primaria de la tabla.

- Junta por único loop: sabiendo que se tiene un índice secundario en la tabla de Multas por *DNI_propietario*, el costo se puede expresar como:

$$C_2 = n_1 * (H(I(DNI_proprietario, Multas)) + \lceil \frac{n(\text{Multas})}{V(DNI_proprietario, Multas)} \rceil)$$

$$C_2 = 1 * (4 + \lceil \frac{60,000}{15,000} \rceil) = 4$$

- El costo de la última selección sería nulo, suponiendo que se encuentra en pipeline:

$$C_3 = 0$$

Luego, el costo total es:

$$C_T = C_1 + C_2 + C_3 = 5,004$$

2.2. Ejercicio 2 (Procesamiento de consultas)

La cantidad de tuplas resultante va a ser ($k = 5$)

	Santa Fe	Bs.As.	Córdoba	Río Negro	Entre Ríos	Otras	sum: 655
Asesores.prov_asignada.	90	110	70	60	45	280	sum: 655
Productores.prov_origen.	1.700	2.200	1.100	640	720	3.100	sum: 9460
Totales parciales (R*S)	153.000	242.000	77.000	38.400	32.400	42.223	

Por lo que la cantidad total de tuplas será $n_T = 585,023$.

Y la estimación de bloques final será

$$B(\bowtie) = \frac{js.n(\text{Asesores}).n(\text{Productores})}{F(R * S)}$$

con $F(R * S) = (\frac{1}{F(\text{Asesores})} + \frac{1}{F(\text{Productores})})^{-1} = 5$.

$$B = \frac{js.n(\text{Asesores}).n(\text{Productores})}{F(R * S)} = \frac{9460 * 655}{23 * 5} = 53881$$

$$js = 1 / (\max(V(R), V(S)))$$

2.3. Ejercicio 3 (NoSQL)

Respuesta

a) { \$match: { "place.country": "Argentina" } },
 { \$unwind: "\$entities.hashtags" },
 { \$group: { _id: "\$entities.hashtags", usos: { \$sum: 1 } } },
 {\$sort:{"usos": -1}},
 {\$limit:5}

Dado que se quiere paralelizar, no se podría shardear por país, porque todos los tweets de Argentina caerían en un mismo nodo. Tampoco sería correcto hashear por arrays, ya que no es algo permitido. Dicho esto, se podría shardear por la fecha de creación o el user.id, teniendo en cuenta que ambos resultarán en gran fraccionamiento de la base. Una vez elegido un modo de hashear que no sea los nombrados arriba, un índice por país se aprovecharía en cada nodo para no tener que leer todos los documentos del nodo.

Si no se quisiera paralelizar, se podría shardear por país, teniendo como ventaja que no habría que totalizar por país para el resultado final. También sería viable shardear por usuario, método en el cual podría venir el mismo país en distintos nodos con lo que habría que hacer un total general de lo que viene.

b) Hay dos opciones

I) MATCH (b:Bodega),
(v:Vino),
(c1:Cepa{nombre: 'Syrah'}),
(c2:Cepa),
b-[:FABRICA]->v-[:ELABORADO_CON]->c
WHERE NOT EXISTS(
MATCH b-[:FABRICA]->v-[:ELABORADO_CON]->c2
WHERE c2<>c1
RETURN b,v
)
RETURN b.nombre, v.nombre, v.precio
ORDER BY v.puntaje
LIMIT 1

II) MATCH (b:Bodega)-[:FABRICA]->(v:Vino)-[:ELABORADO_CON]->(c:Cepa{nombre: 'Syrah'})
WITH b, v, COUNT(DISTINCT c) AS cant_cepas
WHERE cant_cepas=1
RETURN b.nombre, v.nombre, v.precio
ORDER BY v.puntaje
LIMIT 1

2.4. Ejercicio 4 (CRT)

Respuesta

Encuentre los números de licencia de los conductores que tienen al menos dos multas impagadas.

$$\{ l.nrolicencia | \begin{aligned} & Licencias(l) \wedge \\ & (\exists m_1)(Multas(m_1) \wedge m_1.DNI_propietario = l.DNI_conductor \wedge \\ & (\exists m_2)(Multas(m_2) \wedge m_2.DNI_propietario = m_1.DNI_propietario \wedge \\ & m_1.pagada = FALSE \wedge \\ & m_2.pagada = FALSE \wedge \\ & m_1.fecha \neq m_2.fecha)) \end{aligned} \}$$

2.5. Ejercicio 5 (Concurrencia y Transacciones)

a) Respuesta

Que sea recuperable no implica que las transacciones **no** puedan finalizar correctamente.

b) **Respuesta**

Nuevamente, si no hay ABORTs en las transacciones y finalizan correctamente, la anomalía puede suceder.

c) **Respuesta**

Para cualquiera de las variantes implicaría que sucedió

$L_{T_1}(X), \dots, R_{T_1}(X), \dots, U_{T_1}(X), L_{T_2}(X), \dots, W_{T_2}(X), \dots, U_{T_2}(X), L_{T_1}(X), R_{T_1}(X) \dots$

violando así el protocolo

d) **Respuesta**

En SQL, el *Read Committed* evita la anomalía de lectura sucia, pero no impide que suceda la anomalía de lectura no repetible.

Nivel de aislamiento	Lectura sucia	Lectura no repetible	Fantasma
READ UNCOMMITTED	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
READ COMMITTED	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
REPEATABLE READ	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
SERIALIZABLE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

1.6. Ejercicio 6 (Recuperación)

Respuesta

Como lo primero que encuentra el algoritmo al recorrer hacia atrás el *log* es un END CKPT entonces sólo deberá retroceder hasta su BEGIN CKPT .

Luego vemos que T_2 commiteó pero T_3 y T_4 no, por lo que hay que deshacer las instrucciones de T_3 y T_4 en disco, escribiendo:

1. C=9 por T_4 línea 14
2. B=8 por T_3 línea 13
3. A=8 por T_3 línea 08

Finalmente se escribe en el log las instrucciones (ABORT, T_3) y (ABORT, T_4) y se vuelca a disco.

3. Coloquio 2023-12-13

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 13 de diciembre de 2023

TEMA 20232C1				Padrón: _____
SQL	Proc.	NoSQL		Apellido: _____
NoSQL	CyT	Rec.		Nombre: _____
				Cantidad de hojas: _____
				<input type="checkbox"/> Aprobado <input type="checkbox"/> Insuficiente
Nota:				

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 *Aeropuerto de Ezeiza* la torre de control mantiene una tabla con los horarios de reserva de pista de cada vuelo del día, que posee la siguiente estructura:

- Vuelos(*cod_vuelo*, *nro_pista_asignada*, *hora_inicio_pista*, *hora_fin_pista*)

Escriba una consulta en SQL que chequee que no existan dos vuelos de ese día que se superpongan en una misma pista. Si existen vuelos superpuestos, la consulta deberá devolver el par de códigos de vuelos que se superponen y el número de pista que tienen asignada.

2. (*Procesamiento de Consultas*) La empresa de servicios de Internet y televisión *DadaNet* quiere determinar quiénes de sus clientes tienen facturas impagadas 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 esta 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.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

3

4. (*NoSQL*) La *Facultad de Ingeniería* posee una base de datos en MongoDB que almacena cómo está constituido el plantel docente de cada materia. El siguiente documento JSON ejemplifica la estructura almacenada para la materia 78.22 - INTRODUCCION A LOS SISTEMAS DE TRANSPORTE:

```
1 { "cod_dpto": "78",
2   "nombre_dpto": "TRANSPORTE",
3   "subcod_materia": "22",
4   "nombre_materia": "INTRODUCCION A LOS SISTEMAS DE TRANSPORTE",
5   "plantel": [
6     { "legajo": 177815,
7       "apell_nombre": "AUZQUI, BELEN",
8       "cargo": "PROFESOR ADJUNTO"
9     },
10    { "legajo": 201712,
11      "apell_nombre": "ROMAN, HERNAN",
12      "cargo": "JEFE DE TRABAJOS PRACTICOS"
13    },
14    { "legajo": 181218,
15      "apell_nombre": "BERNA, RAUL",
16      "cargo": "AYUDANTE PRIMERO"
17    },
18    { "legajo": 192990,
19      "apell_nombre": "RAMIREZ, GONZALO",
20      "cargo": "AYUDANTE PRIMERO"
21  }
22 }
```

La facultad quisiera conocer quiénes son los docentes que enseñan en materias de más de un departamento. Escriba una consulta en MongoDB que permita resolver esta pregunta, devolviendo el legajo y nombre del docente, y el listado de departamentos en los que enseña (conteniendo código del departamento, nombre del departamento, y cantidad de materias que enseña en ese departamento) sólo para aquellos docentes que enseñan en más de un departamento. El formato debería ser similar al mostrado en el siguiente documento para la docente BELÉN AUZQUI, quien enseña dos materias en el Departamento de Transporte y una materia en el Departamento de Matemática:

```
1 { "_id": {
2   "legajo": 177815,
3   "apell_nombre": "AUZQUI, BELEN"
4 }
5   "listado_departamentos": [
6     {
7       "cod_dpto": "78",
8       "nombre_dpto": "TRANSPORTE",
9       "cant_materias": 2
10    },
11    {
12      "cod_dpto": "81",
13      "nombre_dpto": "MATEMATICA",
14      "cant_materias": 1
15    }
16 }
```

Puede resultarle útil el comando *addToSet* que permite agregar elementos a un conjunto. El

- (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	

3. (*NoSQL*) Ananá Carreira quiere aumentar las ventas de sus famosas pizzas de ananá y jamón. Para ello busca clientes prospectos a los cuales les gusten los platos agridulces.

Afortunadamente para ella, encontró una base de datos en Neo4j de la aplicación *Recetagram*, que cuenta con nodos de tipo USER (que representan usuarios), de tipo DISH (que representan platos), y de tipo INGREDIENT (que representan ingredientes). Los 3 tipos de nodo se identifican por un id y tienen atributos descriptivos como se muestra a continuación:

```
1 {u:USER { id: '0540011052'; email: 'alopez@fi.uba.ar', apellido: 'López',
2   nombre: 'Augusto' })
3 {d:DISH { id: '033895019'; name: 'Pollo a la libanesa' })
4 {i:INGREDIENT { id: '0440195'; name: 'Dátil'; type: 'SWEET' )}
```

El atributo type de los ingredientes puede tomar distintos valores, entre los cuales se encuentran SWEET y SALTY, según si el ingrediente es dulce o salado, respectivamente. Los platos se vinculan con los ingredientes con un arco de tipo USES, mientras que los usuarios pueden puntuar distintos platos, creándose arcos de tipo RATES, como se muestra a continuación:

```
1 (d)-[:USES]->(i);
2
3 (u)-[:RATES {rating: 3}]->(d);
```

El atributo rating del arco RATES toma valores entre 1 y 5, indicando si el plato le gustó mucho (5) o nada (1).

Desafortunadamente, Ananá Carreira no conoce el lenguaje Cypher. Ayúdela a encontrar usuarios prospectos, escribiendo una consulta en Cypher que devuelva el e-mail de aquellos usuarios que hayan puesto al menos 5 platos agridulces con un rating de 3 o más, entendiéndose como un plato agridulce aquel que utiliza tanto ingredientes salados como dulces.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

4

siguiente ejemplo muestra el uso de este comando para encontrar todas las asignaturas de cada departamento:

```
{ $group: { _id: {"cod_dpto": "$cod_dpto", "nombre_dpto": "$nombre_dpto"}, "listado_materias_dpto": {
  $addToSet: {
    "subcod_materia": "$subcod_materia",
    "nombre_materia": "$nombre_materia"
  }
}}
```

5. (*Concurrencia y Transacciones*) ¿Es posible que en un solapamiento de transacciones que respeta el *Protocolo de Lock de Dos Fases (2PL)* el *abort* de una transacción produzca *rollbacks* en cascada de otras transacciones? Justifique su respuesta.

6. (*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, X, 4);
03 (BEGIN, T2);
04 (WRITE, T2, Y, 7);
05 (WRITE, T2, Z, 20);
06 (COMMIT, T1);
07 (BEGIN CKPT, T2);
08 (BEGIN, T3);
09 (WRITE, T3, X, 12);
10 (COMMIT T2);
11 (WRITE, T3, Z, 5);
12 (END CKPT);
13 (BEGIN, T4);
14 (WRITE, T4, Y, 16);
15 (COMMIT, T4);
```

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

12

3.1. Ejercicio 1 (SQL)

Respuesta

```
SELECT v1.cod_vuelo, v2.cod_vuelo, v1.nro_pista_asignada,v2.nro_pista_asignada,
       EXTRACT(DOY FROM v1.hora_inicio_pista) as dia_vuelo_1 ,
       EXTRACT(DOY FROM v2.hora_inicio_pista) as dia_vuelo_2
  FROM Vuelos v1 INNER JOIN Vuelos v2 USING (nro_pista_asignada)
 WHERE EXTRACT(DOY FROM v1.hora_inicio_pista)= EXTRACT(DOY FROM v2.hora_inicio_pista)
   AND v1.cod_vuelo <> v2.cod_vuelo
```

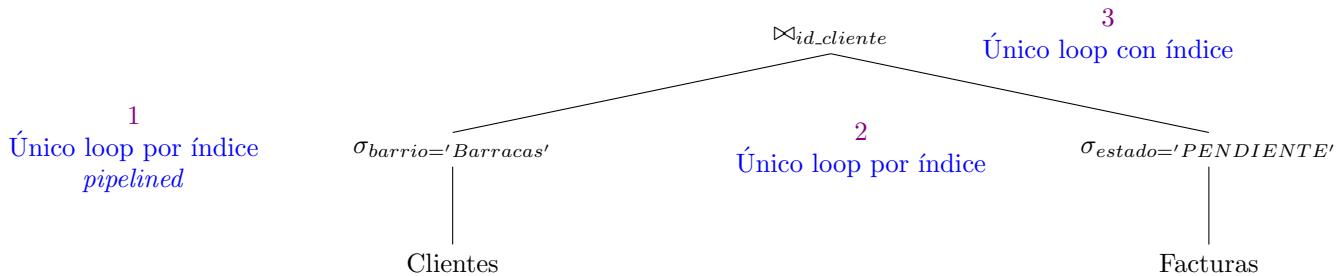
Otra forma sería:

```
SELECT *
  FROM Vuelos v1
 WHERE EXISTS (
    SELECT *
      FROM Vuelos v2
     WHERE EXTRACT(DOY FROM v1.hora_inicio_pista)=EXTRACT(DOY FROM v2.hora_inicio_pista)
       AND v1.cod_vuelo<>v2.cod_vuelo
       AND v1.nro_pista_asignada=v2.nro_pista_asignada
  )
 ORDER BY v1.nro_pista_asignada;
```

Sin bien esta solución encuentra los vuelos que coinciden en día y pista, no se puede devolver el par de códigos de vuelos que se superponen.

3.2. Ejercicio 2 (Procesamiento de consultas)

Respuesta



- La selección del barrio, denotada por 1, al ser una búsqueda por índice de clustering, se tiene que

$$C_1 = H(I(\text{barrio}, \text{Clientes})) + \lceil \frac{B(\text{Clientes})}{V(\text{barrio}, \text{Clientes})} \rceil = 2 + \lceil \frac{10,000}{40} \rceil = 252$$

Height

Y el resultado consta de $\frac{n(\text{Clientes})}{V(\text{barrio}, \text{Clientes})} = 2,500$ filas que entran en $B_1 = 250$ bloques.

- Para calcular el costo de 2, se tiene un proceso similar al anterior, sólo que en este caso el índice es secundario

$$C_2 = H(I(\text{estado}, \text{Facturas})) + \lceil \frac{n(\text{Facturas})}{V(\text{estado}, \text{Facturas})} \rceil = 2 + \lceil \frac{500,000}{4} \rceil = 125,002$$

En este paso, la cantidad de tuplas resultante es $\frac{n(\text{Facturas})}{V(\text{estado}, \text{Facturas})} = 125,000$

- Considerando que sólo hay 500 bloques de memoria y que sólo alcanza para guardar la tabla del paso 1; considerando memoria en disco ilimitada, al resultado del paso 2 habría que guardarlo en disco (verificar) y luego recuperar esos datos.

Dicho esto, y considerando que el índice de `id_cliente` en la tabla `Facturas` es de clustering, el costo de la junta sería:

$$C_3 = n(1) * (H(I(\text{id_cliente}, \text{Facturas})) + \lceil \frac{B(2)}{V(\text{id_cliente}, \text{Facturas})} \rceil) = 2,500 * (3 + \lceil \frac{12,500}{100,000} \rceil) = 10,000$$

La cantidad de tuplas resultante de la junta es $n(3) = \frac{125,000 * 2,500}{100,000} = 3,125$.

Luego, el costo total es:

$$C_T = C_1 + C_2 + C_3$$
$$C_T = 252 + 125,002 + 10,000 = 135,254$$

3.3. Ejercicio 3 (NoSQL)

Respuesta

```
MATCH (u:USER),
      (d:DISH),
      (i1:INGREDIENT{type: 'SWEET'}),
      (i2:INGREDIENT{type: 'SALTY'}),
      u->[:RATES{rating>=3}]->d,
      d->[:USES]->i1,
      d->[:USES]->i2
WHERE COUNT(d)>=5
RETURN u.email
```

3.4. Ejercicio 4 (NoSQL)

Respuesta

```
{ $unwind: "$plantel", desarmo el vector plantel para que todos tengan toda la info
  { $group: { _id: { legajo: "$plantel.legajo", nombre: "$plantel.apell_nombre", cod_dpto: "cod_dpto", nombre_dpto: "nombre_dpto" }, cant_materias: { $sum: 1 } } },
  {
    $group: { _id: { legajo: "$_id.legajo", nombre: "$_id.nombre" }, listado_departamentos: { $push: { cod_dpto: "$_id.cod_dpto", nombre_dpto: "$_id.nombre_dpto", cant_materias: "$cant_materias" } } },
    {
      $match: {
        "$expr": { $gte: [ { $size: "$listado_departamentos" }, 2 ] }
      }
    }
  }
}
```

Nota: Esta solución es incompleta. faltaría agregar la cantidad de materias por departamento que da cada profesor.

3.5. Ejercicio 5 (Concurrencia y Transacciones)

Respuesta

Supongamos que **no** se producen rollbacks en cascada en un solapamiento de transacciones que respetan 2PL. Con un contraejemplo se puede demostrar que es falso.

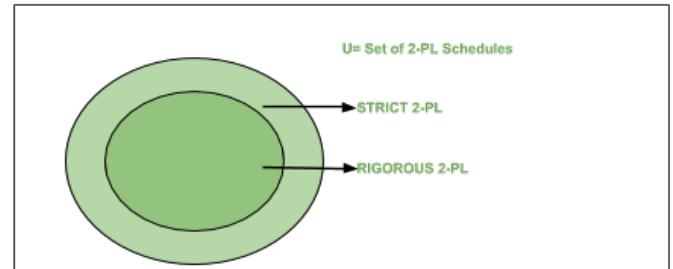
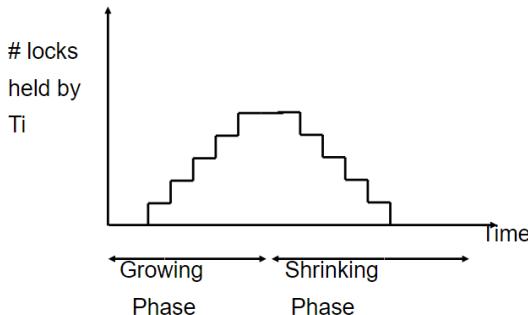
$b_{T_1}; b_{T_2}; L_{T_1}(X); W_{T_1}(X); L_{T_1}(Y); W_{T_1}(Y); U_{T_1}(X); L_{T_2}(X); R_{T_2}(X); ABORT(T_1);$

Recordando que para evitar los rollbacks en cascada es necesario que una transacción no lea valores que aún no fueron commiteados, es lo que no sucede en este caso: T_2 está leyendo el elemento X que fue escrito por T_1 sin que éste haya sido commitado. Por ende, decir que **no** produce rollbacks en cascada es falso.

Otra manera de pensarlo es pensar en el sopalamiento de transacciones bajo el protocolo 2PL de manera gráfica:



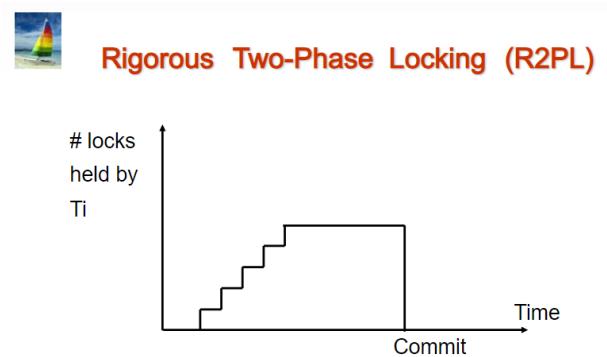
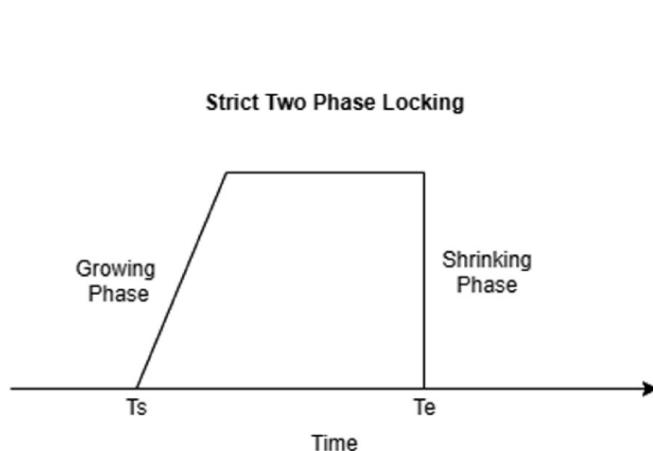
Two-Phase Locking



4

Como el commit de T_i sucede al final de la transacción, si otra transacción T_j toma lock inmediatamente después de que T_i lo libere y, luego T_j lee el elemento y T_i aborta, se daría rollback en cascada de T_i, T_j .

En cambio en R2PL y S2PL no sucedería, porque toda transacción T_i adquiere el lock sobre los elementos que necesita y los libera únicamente luego de commitear, haciendo imposible que otra transacción T_j lea esos elementos al mismo tiempo que T_i .



- Strict 2PL unlock x-locks only at the end of transaction
- Rigorous 2PL releases X-locks & S-locks only at the end of transaction
- As such they exclude dirty reads and produce recoverable cascadeless schedules.

3.6. Ejercicio 6 (Recuperación)

Respuesta

Como lo primero que encuentra el algoritmo, al buscar instrucciones del tipo END CKPT o BEGIN CKPT, es un END CKPT entonces va hasta la última transacción activa al comenzar el checkpoint y rehace las que hayan commiteado, escribiendo también ABORT en las que no hayan commiteado.

Luego vemos que T_1 , T_2 y T_4 commitearon. En particular, T_1 lo hizo antes del END CKPT, por lo que sus modificaciones ya se encuentran en disco, pero las de T_2 y T_4 no. Entonces hay que rehacer las instrucciones de T_2 y T_4 en disco, escribiendo:

1. Y=7 por T_2 línea 04
2. Z=20 por T_2 línea 05
3. Y=16 por T_4 línea 14

Finalmente se escribe en el log (ABORT, T_3) y se vuelca a disco.

T3 nunca se commiteo entonces es ignorada

4. Coloquio 2023-08-02

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 2 de agosto de 2023

TEMA 20231C5				
SQL	Proc.	NoSQL		
NoSQL	CRT	CyT		
Nota: <input type="checkbox"/> Aprobado <input type="checkbox"/> Insuficiente				
Padrón: _____ Apellido: _____ Nombre: _____ Cantidad de hojas: _____				

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) La *Secretaría de Parques Nacionales* mantiene una base de datos relacional con información sobre las distintas reservas del país, los ejemplares de especies que cada una de ellas posee, y si dichas especies se encuentran en peligro de extinción o no.

- Reservas(nombre_reserva, superficie, año_creación)
- Ejemplares(nombre_reserva, nombre_especie, cantidad_ejemplares)
- Especies(nombre_especie, familia, en_extinción)

La Secretaría mide la criticidad de una reserva como la cantidad de especies en peligro de extinción que sólo ella alberga (es decir, la cantidad de especies que dejarían de estar protegidas si esa reserva no existiera). Escriba una consulta en SQL que calcule para cada reserva su nivel de criticidad, devolviendo el nombre de la reserva y la cantidad de especies en peligro de extinción exclusivas de esa reserva. Asegure que la consulta devuelva también aquellas reservas de criticidad cero.

Nota: Puede asumir que la cantidad de ejemplares en la tabla *Ejemplares* es siempre mayor estricta a cero.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

3

3. (NoSQL) La red social *FilmedIn* es una red de contactos entre personajes del mundo del cine destinada a facilitarles la búsqueda de empleo, y que cuenta con una base de datos en Neo4J sobre la cual los usuarios pueden realizar consultas. En esta base de datos se almacenan tres tipos de nodos: *Personas* (que pueden ser Actores, Directores, Guionistas, etc.), *Producciones* (Series, Películas, etc.) y *Empresas* (Estudios, Productoras, etc.).

```

1 (p1:Persona:Actor {nombre: 'Tom Hawks', fecha_nac: date('1963-02-20')})
2 (p1:Producción:Película {nombre: 'Madness', idioma: 'Inglés', duración: 98})
3 (al:Empresa:Estudio {nombre: 'Pixelles', empleados: 730})
  
```

A su vez se almacenan dos tipos de interrelaciones:

- (p1:Persona)-[:CONOCE {fecha_desde: date('2016-05-15')}]->(p2:Persona)
Indica que p1 y p2 están en contacto.
- (p:Persona)-[:PARTICIPÓ]->(pr:Producción), ó (e:Empresa)-[:PARTICIPÓ]->(pr:Producción)
Indica que la Persona p o la Empresa e participó en la Producción pr.

Tom Hawks está intentando obtener un papel en la próxima película del estudio de animación *Pixelles*, y acaba de ejecutar la siguiente consulta en *Neo4J* para averiguar si alguno de sus contactos *c1* participó en alguna producción de dicho estudio:

```

1 MATCH p=(yo: Actor {nombre: 'Tom Hawks'})-[:CONOCE]-
2   (c1: Persona)-[:PARTICIPÓ]->(pr: Producción)<-[ :PARTICIPÓ]-
3   (e: Estudio {nombre: 'Pixelles'})
4 RETURN c1.nombre;
  
```

Lamentablemente esta consulta devolvió un resultado vacío, por lo que Tom intentará ahora buscar a algún contacto suyo *c1* que a su vez tenga un contacto *c2* que haya participado en una producción de *Pixelles*. Dado que Tom posee 300 contactos, espera que esta consulta devuelva varios resultados, y por lo tanto quisiera ordenar a los contactos *c1* en forma descendente por la cantidad de contactos distintos *c2* de cada uno que hayan participado en producciones de *Pixelles*.

Escriba una consulta que devuelva dichos nombres de contacto *c1* y la cantidad de contactos que cada uno tiene que hayan participado en producciones de *Pixelles*, ordenados en forma descendente por este último campo.

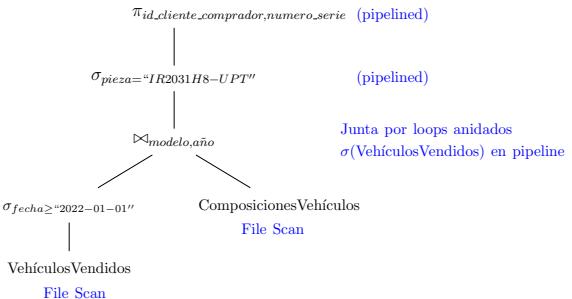
4. (NoSQL) Explique cuál es la importancia de normalizar una base de datos relacional, explicando los problemas que ocasionaría almacenar datos no normalizados. Luego explique brevemente por qué motivo las bases de datos NoSQL como Cassandra o MongoDB se rigen en cambio por el principio de almacenar los datos desnormalizados.

2. (Procesamiento de Consultas) La fabricante de vehículos belga *Bruggeot* ha detectado una falla en el sistema de producción de la pieza de freno 'IR2031H8-UP7', y necesita contactar a quienes hayan comprado vehículos con esa pieza desde el 2022 para que envíen su auto a una revisión gratuita. Para esta tarea, se dispone de las siguientes tablas:

- VehículosVendidos(número_serie, modelo, año, id_cliente_comprador, fecha_venta)
- ComposicionesVehículos(modelo, año, cod_pieza, cantidad)

En donde la primera tabla indica de qué modelo y año era cada vehículo vendido y a qué cliente se vendió, mientras que la segunda tabla indica qué piezas están presentes en cada vehículo de un determinado modelo y año de fabricación.

Para encontrar a los clientes que compraron algún vehículo con la pieza de freno fallada desde el 2022, se arma el siguiente plan de consulta:



Para ejecutar la consulta se dispone de M=2000 bloques de memoria disponibles, y se conoce la siguiente información de catálogo:

VEHICULOS_VENDIDOS	COMPOSICIONES_VEHICULOS
n(VehículosVendidos) = 1.000.000	n(ComposicionesVehículos) = 20.000
B(VehículosVendidos) = 100.000	B(ComposicionesVehículos) = 2.000
Histograma sobre año:	V(modelo, ComposicionesVehículos) = 20
2022: 50.000	V(año, ComposicionesVehículos) = 10
2021: 48.000	V(cod_pieza, ComposicionesVehículos) = 500
2018: 42.000	
2023: 20.000	
otros: 840.000	

Se pide:

- Estime el costo del plan de ejecución anterior en términos de cantidad de bloques de I/O a disco, considerando que dispone de M=2000 bloques de memoria.
- Estime la cardinalidad de la relación resultante, en términos de cantidad de tuplas.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires) 4

5. (Cálculo Relacional de Tuplas) La agencia de viajes *Terramar* se encuentra en la búsqueda de candidatos al nuevo puesto de "Planificador de Paseos" que se ocupará de asistir a los viajeros en las distintas ciudades del mundo que visiten, sugiriéndoles atracciones a visitar y delineando recorridos. La agencia ofrece a los viajeros una serie de paquetes turísticos en los que se recorren distintas ciudades, tal como se representa en la siguiente tabla:

- ComposicionesPaquetes(cod_paquete, ciudad, cantidad_días)
//('AFRICA_SALVAJE', 'Johannesburgo', 3)

Por otra parte, cada candidato sólo conoce algunas de las ciudades ofrecidas por la agencia. Las ciudades que cada candidato conoce y sobre las cuales estaría listo para asistir fue extraída de sus currículums y volcada en la siguiente tabla:

- CiudadesCandidatos(nombre_candidato, ciudad_conocida)
//('Joaquín Kaplan', 'Amsterdam')

La agencia quisiera seleccionar a 3 candidatos que cumplan con la restricción de que cada ciudad incluida en algún paquete de la agencia sea conocida por al menos un candidato, de manera que siempre haya alguien disponible para ayudar sobre cualquier ciudad ofrecida.

Escriba una expresión en Cálculo Relacional de Tuplas que encuentre los nombres (n_1, n_2, n_3) de aquellos candidatos que cumplen con la restricción de que al menos un candidato de entre los tres conozca cada ciudad que la agencia ofrece como destino.

6. (Concurrencia y Transacciones) Dado el siguiente solapamiento de transacciones:

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

- Dibuje el grafo de precedencias del solapamiento.
- Indique si el solapamiento es serializable. Justifique su respuesta.
- Indique si el solapamiento es recuperable. Justifique su respuesta.

4.1. Ejercicio 1 (SQL)

Respuesta

Quiero las reservas que tengan especies en extinción que NO tengan otras reservas, y contar cuántas especies resultan.

```
WITH especies_unicas
AS (
    SELECT *
    FROM Ejemplares ej INNER JOIN Especies es USING (nombre_especie)
    WHERE es.en_extincion= TRUE
        AND NOT EXISTS ( SELECT 1
                            FROM Ejemplares ej2 INNER JOIN Especies es2 USING (nombre_especie)
                            WHERE ej.nombre_reserva <> ej2.nombre_reserva
                                AND ej.nombre_especie = ej2.nombre_especie))
SELECT especies_unicas.nombre_reserva, COUNT(especies_unicas.nombre_reserva) as criticidad
FROM especies_unicas
GROUP BY especies_unicas.nombre_reserva
```

4.2. Ejercicio 2 (Procesamiento de consultas)

Respuesta En cada bloque entrar n/B de lo mismo

$$n(\text{VehVen})/B(\text{VehVen}) = \frac{1.000.000}{100000} = 10 \Rightarrow \text{por bloque entran } 10$$

1. El costo de la primer selección (por año), al ser un simple *file scan* sobre la tabla de *VehiculosVendidos*, es la cantidad de bloques de dicha tabla

$$C_1 = B(\text{VehiculosVendidos}) = 100,000 \quad (2)$$

Con la información del histograma se puede resaltar que la cantidad de tuplas resultantes de este paso son $n_1 = 70,000$ que entran en $B_1 = 7,000$ bloques.

2. En el segundo paso, la junta, al utilizar el método de junta de loops anidados por bloque con $M = 2,000$ bloques de memoria disponibles, tendría un costo de

$$C_2 = [B(1)/(2,000 - 2)] * B(\text{ComposicionesVehiculos}) = 8,000 \quad (3)$$

Para verificar la cantidad de tuplas resultante de la juntase puede considerar que al tener 20 modelos y 10 años, se tiene como máximo una variedad de 200 autos $\Rightarrow V(\text{modelo-año}, \text{ComposicionesVehiculos}) = 200$. Luego, la cantidad de tuplas se calcula como:

$$n(2) = \frac{n(1) * n(\text{ComposicionesVehiculos})}{V(\text{modelo-año}, \text{ComposicionesVehiculos})} = \frac{70,000 * 20,000}{200} = 7,000,000 \quad (4)$$

3. Los siguientes pasos no agregan costos de accesos a disco, al estar en *pipeline*.

En cuanto a la cantidad de tuplas de estas etapas: la selección reduce en $\frac{1}{500}$ el resultado previo por la selección, y supongo que todas las tuplas resultantes son de clientes distintos, por lo que no habría que eliminar duplicados. Entonces:

$$n(3) = \frac{7,000,000}{500} = 14,000 \quad (5)$$

Luego, el costo total es:

$$C_T = C_1 + C_2 + C_3 \\ C_T = 100,000 + 8,000 + 0 = 108,000$$

4.3. Ejercicio 3 (NoSQL)

Respuesta

```
MATCH p=(yo: Actor {nombre: 'Tom Hawks '})-[:CONOCE]-
(c1: Persona)-[:CONOCE]-(c2: Persona)-[:PARTICIPO]->(pr: Produccion)<-[:PARTICIPO]-
(e: Estudio {nombre: 'Pixelles '})
RETURN c1.nombre, size((c1)-[:CONOCE]-(c2)-[:PARTICIPO]->(pr)<-[:PARTICIPO]-(e)) AS cant_contactos
ORDER BY cant_contactos DESC
```

4.4. Ejercicio 4 (NoSQL)

Respuesta

En una base de datos relacional se requiere normalizar para reservar la información eliminar la redundancia, reducir la probabilidad de ocurrencia de anomalías de ABM, todo mientras se preserva la información de la base de datos.

Sin embargo, en las bases de datos NoSQL se desea almacenar datos desnormalizados para priorizar la simplicidad y la dinámica cambiante de los datos. Ellas permiten mayor flexibilidad sobre las estructuras de datos, permitiendo que evolucionen en el tiempo; mayor capacidad de distribución, buscando mayor disponibilidad y tolerancia a fallas de parte del SGBD; y mayor escalabilidad, para trabajar con grandes volúmenes de datos.

En Cassandra por ejemplo podría tenerse una base de datos que almacena mails de personas: algunas tienen un mail, mientras que otras pueden tener n . La idea es que pueda variar para cada individuo. Bajo un esquema relacional normalizado, esto no sería ni simple ni conveniente, pero con la estructura de *wide rows* de Cassandra, se podrían agregar a medida. Lo mismo sucedería con Mongo: al ser una base no relacional orientada a documentos, una persona podría tener un vector de documentos a medida.

4.5. Ejercicio 5 (CRT)

Respuesta

Quiero "La agencia quisiera seleccionar a 3 candidatos que cumplan con la restricción de que cada ciudad incluída en algún paquete de la agencia sea conocida por al menos un candidato, de manera que siempre haya alguien disponible para ayudar sobre cualquier ciudad ofrecida."

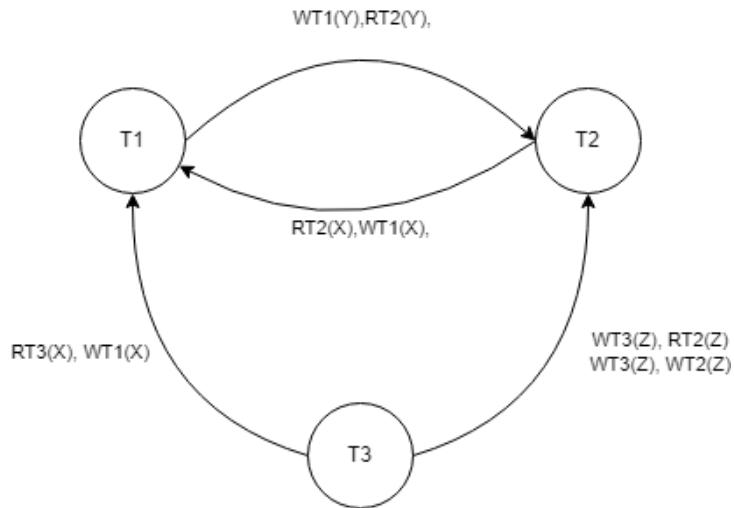
Dicho de otra forma, quiero tres candidatos **distintos** tales que entre los tres conozcan todas las ciudades de un paquete.

Una primera opción podría ser usando la el \neq :

$$\{n_1.nombre_candidato, \\ n_2.nombre_candidato, \\ n_3.nombre_candidato | \begin{array}{l} n_1, n_2, n_3 \in CiudadesCandidatos \wedge \\ n_1.nombre_candidato \neq n_2.nombre_candidato \wedge \\ n_2.nombre_candidato \neq n_3.nombre_candidato \wedge \\ n_1.nombre_candidato \neq n_3.nombre_candidato \wedge \\ (\forall p \in ComposicionesPaquetes)((\nexists c \in ComposicionesPaquetes)(\\ c.cod_paquete = p.cod_paquete \\ c.ciudad \neq n_1.ciudad_conocida \wedge \\ c.ciudad \neq n_2.ciudad_conocida \wedge \\ c.ciudad \neq n_3.ciudad_conocida)))\}$$

Nota: pero esta opción tiene otro problema, y es que se está limitando a las 3 ciudades que aparecen en n_1 , n_2 , y n_3 , cuando posiblemente esos 3 candidatos conozcan entre ellos muchas más ciudades.

4.6. Ejercicio 6 (Concurrencia y Transacciones)



Como el grafo tiene ciclos, **no** es serializable.

Un solapamiento es recuperable si ninguna transacción T_i commitea hasta que las T_j que escriben elementos antes que T_i las leyera hayan commiteado \Rightarrow no es recuperable porque T_2 commitea antes que T_1 y T_1 escribe datos que T_2 lee.

5. Coloquio 2023-07-26

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 26 de julio de 2023

TEMA 20231C4			Padrón: _____
CRT	Proc.	NoSQL	
NoSQL	CyT	Rec.	
Nota: _____			
<input type="checkbox"/> Aprobado <input type="checkbox"/> 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. (*Cálculo Relacional de Tuplas*) Las siguientes tablas mantienen información sobre las selecciones nacionales femeninas que históricamente han formado parte de la FIFA y las jugadoras que han representado a dichas selecciones. Se quisiera conocer quiénes son las mayores goleadoras de cada selección nacional femenina desde su existencia.

- Selecciones(cod_país, nombre_país, año_creación)
// ('ES', 'España', 1983)
- Jugadoras(cod_jugadora, apellido, nombre, fecha_nacimiento)
// (7142, 'Pablos', 'Natalia', 1985-10-15)
- Representaciones(cod_jugadora, cod_país, cant_partidos_jugados, cant_goles)
// (7142, 'ES', 22, 13)

Escriba una consulta en *Cálculo Relacional de Tuplas* que encuentre para cada selección su/s máxima/s goleadora/s histórica/s, indicando el nombre completo de la selección (p.ej., *España*) y el apellido y nombre de dicha/s jugadora/s.

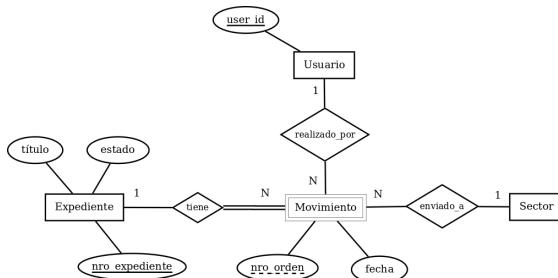
Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

3

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

4

3. (*NoSQL*) El siguiente diagrama muestra la información almacenada en el Sistema de Expedientes de la Facultad, respecto a los movimientos que hace cada expediente a lo largo de su vida. Un expediente es abierto por un usuario en específico, que es quien realiza el movimiento "0" del mismo enviándolo a un primer sector. Luego, en movimientos subsiguientes realizados por distintos usuarios el expediente se mueve de sector, hasta que en algún momento es cerrado. El estado del expediente representa si el mismo se encuentra abierto o cerrado al día de hoy.



Diseñe una familia de columnas en Cassandra que permita consultar un número de expediente determinado y devolver su estado actual y el listado de saltos que hizo por distintos sectores de la Facultad, indicando para cada paso el número de orden, el id del usuario que lo transfirió en esa etapa, la fecha en que se hizo dicha transferencia y el código del sector al que llegó. La familia de columnas también deberá permitir hallar fácilmente el estado del expediente, el último sector en que se encuentra y la fecha en la que llegó a dicho sector. Indique claramente cuáles serían la clave de partición, la clave de clustering y eventuales atributos estáticos de la familia de columnas, ya sea dibujando el diagrama Chebotko o bien escribiendo el CREATE TABLE de la misma.

4. (*NoSQL*) Explique qué problema resuelve el mecanismo de *hashing consistente* en Dynamo, y qué ventajas presenta respecto a un *hashing* módulo N.

2. (*Procesamiento de Consultas*) La plataforma de juegos online Vapor quiere conocer información de los usuarios que poseen juegos de al menos uno de los siguientes dos desarrolladores argentinos: "GAME NEVER" y "4E INTERACTIVE". Para ello, ejecutará la siguiente consulta sobre las dos tablas, que le devolverá los ids de usuarios que poseen dichos juegos:

- Juegos(id_juego, nombre, categoría, desarrollador, fecha_lanzamiento)
- Librerías(id_juego, id_usuario, fecha_compra)

Para encontrar a los usuarios que compraron alguno de esos juegos, se arma el siguiente plan de consulta:



Proponga un plan de ejecución eficiente que respete el plan de consulta anterior, indicando el método utilizado para resolver cada etapa, y estimando su costo. Utilice la siguiente información de catálogo que indica entre otros detalles cuáles son los índices existentes, asumiendo que todos ellos son índices secundarios:

JUEGOS	LIBRERIA
n(Juegos) = 1.000.000	n(Librerías) = 50.000.000
B(Juegos) = 100.000	B(Librerías) = 1.000.000
V(categoría, Juegos) = 50	V(id_juego, Librerías) = 1.000.000
V(desarrollador, Juegos) = 10.000	V(id.usuario, Librerías) = 500.000
H(I(id_juego, Juegos)) = 5	H(I(id_juego, Librerías)) = 6
H(I(categoría, Juegos)) = 4	H(I(id.usuario, Librerías)) = 6
H(I(desarrollador, Juegos)) = 5	H(I(id.usuario, Librerías)) = 6

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

5. (*Concurrencia y Transacciones*) Dado el siguiente par de transacciones:

$$\begin{aligned} T_1 : & b_{T_1}; R_{T_1}(B); W_{T_1}(B); R_{T_1}(A); W_{T_1}(B); c_{T_1}; \\ T_2 : & b_{T_2}; R_{T_2}(A); R_{T_2}(B); W_{T_2}(B); R_{T_2}(C); c_{T_2}; \end{aligned}$$

Se pide:

- Coloque *locks* y *unlocks* a ambas transacciones de manera de respetar el *Protocolo de Lock de 2 Fases*, intentando a la vez minimizar el tiempo que las transacciones mantienen los *locks* sobre los recursos.
- Intente proponer un solapamiento que respete esos *locks* y tal que en algún momento las transacciones queden en *deadlock*. Si considera que eso no es factible en este caso, justifique por qué.

6. (*Recuperación*) El gestor de *log* de una base de datos utiliza un *log* de tipo REDO. Indique si cada una de las siguientes afirmaciones sobre el funcionamiento del mismo es verdadera (V) ó falsa (F), justificando su respuesta.

- El registro de *log* correspondiente a un *write.item(X)* debe ser volcado a disco después de volcar el nuevo valor de *X* modificado por dicha instrucción a disco.
- El registro de *log* correspondiente al *commit* de una transacción T_i debe ser volcado a disco después de volcar el nuevo valor de todo ítem *X* modificado por T_i a disco.
- Un *checkpoint* activo recién podrá terminarse (es decir, recién se podrá escribir el *END CKPT* en el *log*) una vez que todas las transacciones que estaban activas al inicio del *checkpoint* hayan terminado y sus datos hayan sido enviados a disco.
- En caso de recuperación, si se debe abortar una transacción T_i que no había terminado, entonces antes de escribir el registro de *log* correspondiente al *abort* de T_i deberán volcarse a disco los valores anteriores *v_{old}* de cada ítem que había sido modificado por T_i .

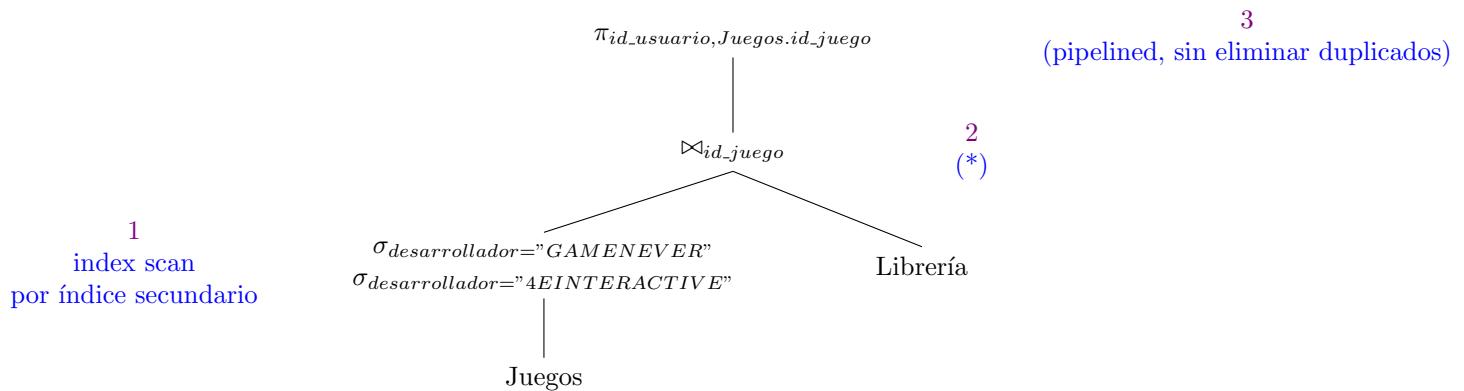
5.1. Ejercicio 1 (CRT)

Respuesta

$$\{s.nombre_pais, j.nombre, j.apellido \mid \text{Jugadoras}(j) \wedge \text{Selecciones}(s) \wedge (\exists r_1)(\text{Representaciones}(r_1) \wedge r_1.cod_pais = s.cod_pais \wedge r_1.cod_jugadora = j.cod_jugadora \wedge (\sim \exists r_2)(\text{Representaciones}(r_2) \wedge r_2.cant_goles > r_1.cant_goles))\}$$

5.2. Ejercicio 2 (Procesamiento de consultas)

Respuesta



- Para el costo de la instrucción 1 es necesario tener en cuenta que existe un índice secundario para buscar por desarrollador en la tabla de Juegos.

Entonces, sabiendo que hay que buscar los datos en disco

$$C'_1 = H(I(\text{desarrollador}, \text{Juegos})) + \lceil \frac{n(\text{Juegos})}{V(\text{desarrollador}, \text{Juegos})} \rceil$$

$$C'_1 = 5 + \lceil \frac{1,000,000}{10,000} \rceil$$

$$C'_1 = 105$$

Esta sería el costo para una búsqueda por igual. Como la búsqueda hay que realizarla para dos desarrolladores distintos, el costo final sería:

$$C_1 = 210$$

Nota: quizás hay una optimización del SGBD para realizar una sola búsqueda y obtener las 200 filas, pero únicamente reduciría en 5 el costo (acceso a los datos mediante el índice secundario), por lo que no se considera relevante.

Se seleccionan 200 filas ($100 = \frac{n(\text{Juegos})}{V(\text{desarrollador}, \text{Juegos})}$ por cada desarrollador que se selecciona), que entran en 20 bloques de memoria (en cada bloque entran $10 = \frac{n(\text{Juegos})}{B(\text{Juegos})}$).

- Para calcular el costo de 2 hay varias opciones.

Se puede asumir pipelining: a medida que las filas seleccionadas van apareciendo, se va procesando la junta. También se podría asumir que el resultado de 1 entra en memoria.

- Junta por loops anidados: como se van procesando las filas a medida que van apareciendo, por el pipelining el costo sería

$$C_{2a} = B(\text{Librerias})$$

$$C_{2a} = 1,000,000$$

ya que sólo se deben ir a buscar los bloques de la tabla de Librerías a disco, y la comparación se hace en memoria. Si la tabla de Librerías no entrase en memoria, pero la de la primera etapa sí (asumiendo que no hay pipelining), se debería traer de a un bloque de la tabla de Librerías y comparar la tabla de la primera etapa contra cada uno de esos bloques, resultando en un costo de $B(1) * B(S)$ (no sería $\min(B(1) + B(1) * B(S), B(S) + B(1) * B(S))$ porque una de las tablas ya está en memoria).

- b) Junta por unico loop: Sabiendo que también se tiene un índice secundario en la tabla de libreras por `id_juego`, el costo anterior se podría expresar como:

$$C_{2b} = n(1) * (H(I(id_juego, Librerias)) + \lceil \frac{n(Librerias)}{V(id_juego, Librerias)} \rceil)$$

$$C_{2b} = 200 * (6 + 50)$$

$$C_{2b} = 11200$$

- c) Junta HASH GRACE: Como una de las tablas del \bowtie_{id_juego} se procesa en pipeline (o ya está en memoria), sólo se tendría el costo de armar las particiones de la tabla **Librerías** en memoria. Éste costo de armado de particiones es de $2B(R)$ para cada tabla R , $\Rightarrow 2B(Libreria)$.

Luego es necesario considerar el costo de comparaciones entre particiones (R_i, S_i), que si ambas están en disco sería $B(R) + B(S)$, pero como una de las tablas ya está en memoria sería únicamente $B(S)$, entonces el costo total sería:

$$C_{2c} = 3 * B(Librerias)$$

$$C_{2c} = 3,000,000$$

Aquí se asume que la tabla **Librerías** no entra en memoria $\Rightarrow M < 1,000,000$. Además la cantidad de particiones debe respetar $k < M, k < V(id_juegos, Librerias) = 1,000,000, k < V(id_juegos, Juegos) = n(Juegos) = 1,000,000$.

Tip: Cuando hay un índice por un atributo de junta en una de las tablas, probar primero con el método de **único loop**, y no usar algo que tiene más accesos a disco (creación de particiones para HASH GRACE).

3. Como en el paso 3 la proyección se hace por una clave del resultado de la junta, no es necesario eliminar duplicados y no cambia el costo (aunque sí la cantidad de bloques del resultado).

Luego, el costo total es:

$$C_T = C_1 + C_{2b} + C_3$$

$$C_T = 210 + 11200 + 0 = 11410$$

5.3. Ejercicio 3 (NoSQL)

Respuesta

Diagrama Chebotko

nro_expediente	uuid	K
estado	text	S
nro_orden	uuid	C ↓
fecha	text	
user_id	text	
cod_sector	text	

Tabla 1: Expedientes

Código CQL

```
CREATE COLUMNFAMILY clientes (
    nro_expediente int,
    estado text static ,
    nro_orden bigint ,
    fecha text ,
    user_id bigint,
    cod_sector int,
    primary key (( nro_expediente ), nro_orden );
    WITH CLUSTERING ORDER BY (nro_orden DESC);
```

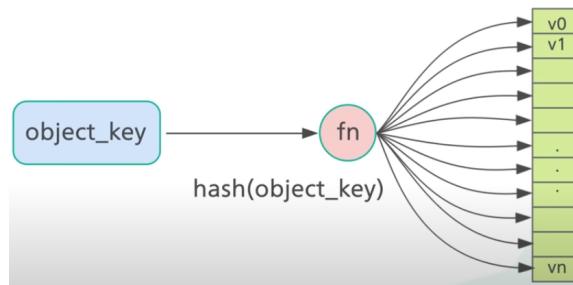
5.4. Ejercicio 4 (NoSQL)

Respuesta

DynamoDB usa el hashing consistente para el lookup (búsqueda de información en nodos). Permite reducir la cantidad de movimientos de pares que se necesitan cuando cambia la cantidad de nodos (servers), haciendo que sea fácil agregar nodos de forma dinámica con impacto mínimo.

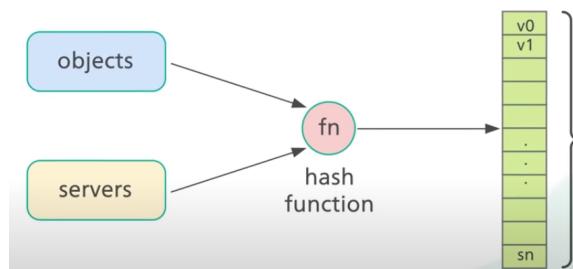
Además es descentralizado, careciendo de un único punto de falla.

Suponiendo x un elemento, el esquema del hash módulo N (siendo N la cantidad de servers y tomando $N = 3$), sería:

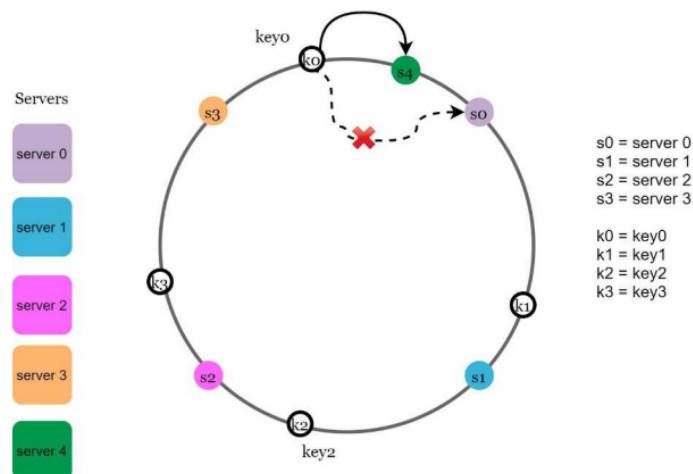


Si se cae un server, por ejemplo S_2 , con el hashing módulo N ahora $N' = 2$ en vez de $N = 3$. Entonces habría que redistribuir todos los elementos de S_2 además de los que habría que reubicar en S_1 y S_3 .

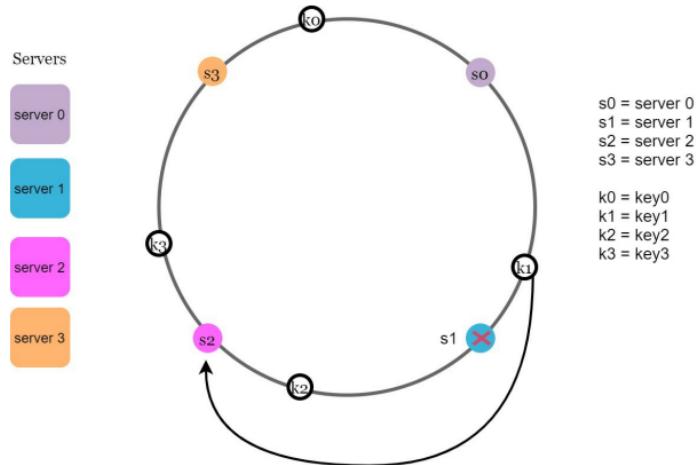
Ahora, teniendo en cuenta el esquema de hashin consistente, también se aplica la función de hash al server



Al disponer los nodos (servers) de una estructura de hash creciente, si se agrega un server (por ejemplo S_4) sólo habría que reubicar las claves 'anteriores' ($1/N$ datos) y no la totalidad de los datos como en el hashing de módulo N .



Lo mismo sucede si se cae algún nodo (por ejemplo S_3): sólo las claves previas deberán ser reubicadas. En este ejemplo, K_1 ahora irían a S_2 .



5.5. Ejercicio 5 (Concurrencia y Transacciones)

Dado el siguiente par de transacciones:

$$T_1 : b_{T_1}; R_{T_1}(B); W_{T_1}(B); R_{T_1}(A); W_{T_1}(B); c_{T_1}$$
$$T_2 : b_{T_2}; R_{T_2}(A); R_{T_2}(B); W_{T_2}(B); R_{T_2}(C); c_{T_2}$$

Se pide:

- Coloque locks y unlocks a ambas transacciones de manera de respetar el Protocolo de Lock de 2 Fases, intentando a la vez minimizar el tiempo que las transacciones mantienen los locks sobre los recursos.

Respuesta

T_1	T_2
	b_{T_2}
	$L(A_{sh})$
	$R_{T_2}(A)$
b_{T_1}	
$L(B)$	
$R_{T_1}(B)$	
$W_{T_1}(B)$	
$L(A_{sh})$	
$R_{T_1}(A)$	
$W_{T_1}(B)$	
c_{T_1}	
$U(A_{sh})$	
$U(B)$	
	$L(B)$
	$R_{T_2}(B)$
	$W_{T_2}(B)$
	$L(c)$
	$R_{T_2}(C)$
	c_{T_2}
	$U(A_{sh})$
	$U(B)$
	$U(C)$

- Intente proponer un solapamiento que respete esos locks y tal que en algún momento las transacciones queden en deadlock. Si considera que eso no es factible en este caso, justifique por qué.

Respuesta

5.6. Ejercicio 6 (Recuperación)

a) Respuesta

F En **ningún** algoritmo de recuperación se reliza el *flush* del ítem a disco y luego el *flush* del *log* de esa instrucción a disco. Se quiere tener la información del *log* en disco **siempre** antes de realizar modificaciones en ítems en disco. Tanto REDO como UNDO siguen la regla WAL (*Write Ahead Log*), que menciona el flush a disco de los ítems **luego** de haber hecho el flush a disco del *log* de esos ítems.

b) Respuesta

F En el algoritmo REDO se hace *flush* de la instrucción del *commit* a disco y luego, en algún momento, se realiza el *flush* de los ítems a disco. De esta manera, si se commiteó la transacción y los ítems no están en disco, habría que reahcer la transacción; si no se commiteó la transacción no se rehace nada y se indica el **ABORT** de esa transacción.

c) Respuesta

F En el algoritmo REDO con checkpoint activo, la instrucción de END CKPT se escribe una vez que las instrucciones de las transacciones ya commiteadas fueron volcadas a disco. No espera a que las transacciones activas t_{act} terminen.

d) Respuesta

F Como los nuevos valores de los ítems son cargados a disco **luego** de haber realizado el *commit* de esa transacción T_i , si sucede algo **antes** del *commit*, T_i no terminó y los nuevos valores los ítems que modificó no fueron cargados a disco, por lo que no habría que volver a cargar sus valores viejos.

6. Coloquio 2023-07-19

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 19 de julio de 2023

TEMA 20231C3				Padrón: _____
Proc.	SQL	NoSQL		Apellido: _____
NoSQL	CyT	CyT		Nombre: _____
				Cantidad de hojas: _____
				<input type="checkbox"/> Aprobado <input type="checkbox"/> Insuficiente
Nota: _____				

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*) Un club de golf mantiene en una base de datos un registro de todos sus hoyos, las personas afiliadas al club, las distintas partidas jugadas entre ellas y la puntuación de cada jugador para cada hoyo en cada una de las partidas:

- Hoyos(nro_hoyo, hándicap, par)
- Socios(nro_socio, apellido, nombre)
- Partidas(cod_partida, fecha, hora)
- JugadoresPartidas(cod_partida, nro_socio)
- Puntuaciones(cod_partida, nro_hoyo, nro_socio, puntuación)

A fin de organizar un torneo especial, el club quiere encontrar el número de socio, apellido y nombre de aquellos/as socios/as que hayan logrado alguna vez completar un hoyo de hándicap entre 1 y 6 en una cantidad de golpes menor o igual al par del hoyo. Para encontrar esta información, se ejecutará la siguiente consulta SQL:

```
SELECT DISTINCT (nro_socio, apellido, nombre)
FROM Socios NATURAL JOIN Puntuaciones NATURAL JOIN Hoyos
WHERE puntuacion <= par
AND handicap <= 6;
```

Se pide:

- Proponga dos índices cuya existencia considere que ayudaría a realizar eficientemente la consulta anterior.
- Diseñe un plan de ejecución para la consulta, empleando los 2 índices que propuso. Estime el costo de la consulta en términos de cantidad de bloques transferidos desde/hacia disco.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

3

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

4

La gerencia identifica a las familias que tienen bebés como aquellas que adquieren pañales "Makako" en sus compras. Recientemente se ejecutó la siguiente consulta para analizar la evolución mensual de la cantidad de compradores únicos que han adquirido dichos pañales:

```
[{"$match": {"productos.codigo_producto": {"$in": [108431]}}}, {"$project": {"mes": {"$month": "$fecha_venta"}, "año": {"$year": "$fecha_venta"}, "comprador": {"$SDNI_comprador"}}, {"$group": {"_id": {"año": "$año", "mes": "$mes"}, "compradores": {"$addToSet": {"$comprador"}}, {"$project": {"cant.compradores": {"$size": "$compradores"}}, {"$sort": {"_id.año": -1, "_id.mes": -1}}, {"$limit": 10}]}]
```

Lamentablemente, esta consulta mostró que en los últimos 10 meses la cantidad de compradores de pañales ha ido en disminución, y la gerencia lo atribuye a que las familias con bebés están frecuentando menos el supermercado.

Se quiere entonces proponer ofertas interesantes para las familias que tienen bebés, y para ello se desea saber cuáles son los productos más frecuentemente adquiridos junto con pañales *Makako*. Escriba una consulta en MongoDB que encuentre los cinco productos que aparecen más frecuentemente en tickets de compra que incluyen pañales *Makako*, indicando el código del producto y la cantidad de tickets de compra distintos en los que aparece combinado con dichos pañales. El formato de los documentos resultantes deberá ser:

```
[{"_id": 210160, "cantidad_tickets_compra": 3804}, ...]
```

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

- En una base de datos distribuida en que nunca se producen particiones es posible en teoría obtener máxima consistencia y disponibilidad simultáneamente.
- DynamoDB prioriza la consistencia por sobre la disponibilidad.
- En Cassandra, el contenido de una *wide row* se almacena entero en un nodo, y puede a la vez estar replicado por enteros en varios nodos.
- MongoDB permite realizar fragmentación vertical, distribuyendo el contenido de un mismo documento entre distintos nodos.
- No es posible implementar transacciones ACID en una base de datos distribuida.

- (c) Indique si la proyección final deberá eliminar duplicados o no. Justifique su respuesta.

Puede asumir que dispone de memoria ilimitada para ejecutar la consulta. Considere para sus cálculos la siguiente información de catálogo:

SOCIOS	PUNTUACIONES	HOYOS
n(Socios) = 20.000	n(Puntuaciones) = 500.000	n(Hoyos) = 30
B(Socios) = 1.000	B(Puntuaciones) = 50.000	B(Hoyos) = 2
V(nro_hoyo, Puntuaciones) = 30	V(nro_socio, Puntuaciones) = 15.000	Histograma sobre hándicap: [1, 6] : 5 [7, 12] : 12 [13, 18] : 13

2. (*SQL*) A partir de las mismas tablas del ejercicio anterior, escriba una consulta en SQL que encuentre el número de socio, apellido y nombre de aquellos/as socios/as que **jamás** hayan completado un hoyo en una cantidad de golpes (es decir, 'puntuación') mayor que la cantidad indicada por el par del hoyo.

3. (*NoSQL II*) Una cadena de supermercados almacena información sobre las ventas que realiza en una base de datos MongoDB. El siguiente documento JSON ejemplifica la estructura almacenada por cada venta, que incluye los siguientes campos:

- El número de ticket, que se utiliza como *_id*.
- La fecha y hora de la venta.
- El DNI del comprador.
- El nombre del comprador.
- Un vector de documentos embebidos conteniendo cada uno un código de producto, nombre de producto, cantidad de unidades y precio por dicha cantidad de unidades.
- El monto total de la venta.

```
[{"_id": "0057-46290015", "fecha_venta": Date("2023-04-19 11:53:28"), "DNI_comprador": 38115218, "nombre_comprador": "BENITEZ, JUAN ANTONIO", "productos": [{"codigo_producto": 108431, "nombre_producto": "pañales makako 20 un.", "cantidad_producto": 3, "precio_por_cantidad": 1048.50}, {"codigo_producto": 210160, "nombre_producto": "papas fritas knix 150 gr.", "cantidad_producto": 1, "precio_por_cantidad": 372.00}], "monto_total": 282.50}]
```

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

4

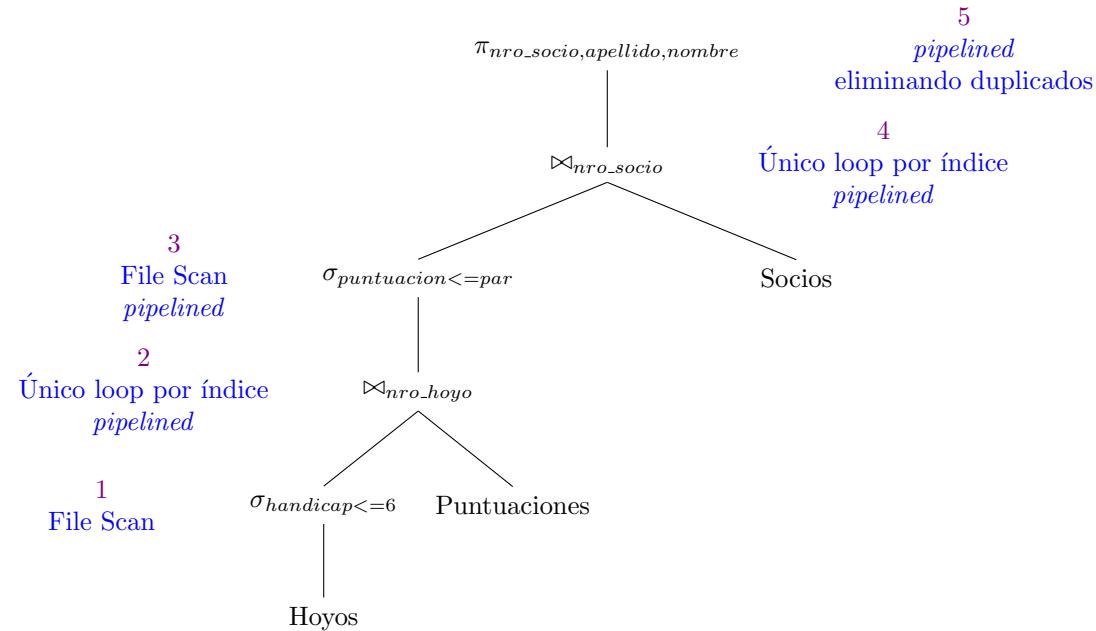
5. (*Concurrencia y Transacciones*) Un SGBD soporta el manejo de transacciones ACID en forma concurrente utilizando el *Protocolo de Lock de 2 Fases (2PL)*. En este contexto, indique si las siguientes afirmaciones relativas a la ejecución de transacciones son verdaderas ó falsas. Justifique cada una de sus respuestas.

- Una transacción T_i no tomará un lock sobre un ítem X hasta tanto toda otra transacción T_j que tomó un lock sobre el ítem X previamente haya hecho su *commit*.
- Una transacción T_i no volverá a tomar un lock sobre un ítem X luego de haberlo liberado.
- Una transacción T_i no tomará un lock sobre un ítem X luego de haber liberado un lock sobre otro ítem Y .
- Una transacción T_i no leerá un ítem X sin tener en posesión un lock sobre dicho ítem.
- Una transacción T_i no escribirá un ítem X sin tener en posesión un lock sobre dicho ítem.
- Una transacción T_i nunca poseerá locks sobre dos ítems distintos, X e Y , simultáneamente.

6. (*Concurrencia y Transacciones II*) Explique brevemente qué ventajas y desventajas tiene el método de control de concurrencia *Snapshot Isolation* por sobre el control basado en *locks*.

6.1. Ejercicio 1 (Procesamiento de Consultas)

Respuesta



- Para el costo de la instrucción 1, al ser simplemente una búsqueda lineal, se tiene que

$$C_1 = B(Hoyos)$$

$$C_1 = 2$$

Y el resultado consta de 5 filas, que están dadas por el histograma; y 1 bloque, pues $\frac{n(Hoyos)}{B(Hoyos)} = 15$.

- Para calcular el costo de 2, suponiendo que hay suficiente espacio en memoria para guardar los resultados del paso anterior, se propone un índice secundario en Puntuaciones usando nro_hoyo de altura 6. Luego, el costo resulta:

$$C_2 = n(1) * (H(I(nro_hoyo, Puntuaciones)) + \frac{n(Puntuaciones)}{V(nro_hoyo, Puntuaciones)})$$

$$C_2 = 5 * (6 + 16,667) = 83,365$$

Si en cambio, el índice fuera de clustering, el costo sería:

$$C'_2 = 5 * (6 + 1667) = 8,365$$

diez veces menor. Sin embargo, no pareciera tener mucho sentido generar un ordenamiento físico en la tabla de Puntuaciones por nro_hoyo dado que ni siquiera forma parte de la clave primaria de la tabla. Se justificaría si hay muchas consultas de ese estilo.

Suponiendo el peor caso, donde cada una de las 5 filas del resultado previo (5 hoyos distintos) se juntan con 16,666 filas de la tabla de Puntuaciones, se tendría $n(2) = 83,334$ (da lo mismo que aplicar la formula $n = js * n(1) * n(Puntuaciones)$) y $B(2) = js * B(1) * B(Puntuaciones) * (F(1) + F(Puntuaciones)) = \frac{1}{20,000} * 1 * 50,000 * (15 + 10) = 41,667$.

- Como el paso 3 está en pipeline, no hay operaciones de acceso a disco, por ende su costo es cero. En el peor caso, todas las filas de la etapa previa se mantienen en esta selección, entonces $n(3) = 83,334$ y $B(3) = 41,667$.

- Usando la misma lógica que en el paso 2, y suponiendo una altura de 5 para este índice, que se toma como primario:

$$C_4 = n(3) * (H(I(nro_socio, Socios)) + 1)$$

$$C_4 = 83,334 * (5 + 1) = 500,000$$

$$B(4) = js * B(4) * B(Socios) * (F(4) + F(Socios)) = \frac{1}{20,000} * 41,667 * 1,000 * (2 + 20) = 45,834$$

- La proyección se hace por atributos que no conforman una clave del resultado de la junta, por lo que sería necesario eliminar duplicados, y habría que sumarle un costo extra, pero no sería costo de accesos a disco, sino computacional. Entonces:

$$C_5 = 0$$

Luego, el costo total es:

$$C_T = C_1 + C_2 + C_3 + C_4 + C_5$$

$$C_T = 2 + 83,365 + 0 + 500,000 + 0 = 583,367$$

El resultado de la proyección es el resultado de dos juntas: la primera entre una tabla de **Hoyos y Puntuaciones**; y luego el resultado de esta primera junta con la tabla de **Socios**. En ese caso, la tabla final tiene como clave al conjunto de (**cod_partida**, **nro_hoyo**, **nro_socio**), y la proyección sólo contiene al atributo de **nro_socio**.

Preguntar resolución y consultar el cálculo de C_5 .

6.2. Ejercicio 2 (SQL)

Respuesta

Ésto se puede pensar como una división usando un doble "NOT EXISTS", con la lógica de "Dame a los socios para los cuales no exista un hoyo que tenga puntuación mayor al par".

```
SELECT DISTINCT s.nro_socio, s.apellido, s.nombre
FROM Socios s
WHERE NOT EXISTS (
    SELECT 1
    FROM Hoyos h
    WHERE EXISTS (
        SELECT 1
        FROM Puntuaciones p
        WHERE p.nro_socio = s.nro_socio
        AND p.nro_hoyo = h.nro_hoyo
        AND p.puntuacion > h.par))
```

6.3. Ejercicio 3 (NoSQL)

Respuesta

```
[
  { $match: { "productos.codigo_producto": { $in: [108431] } } },
  { $unwind: "$productos" },
  { $match: { "productos.codigo_producto": { $ne: 108431 } } },
  {
    $group: {
      _id: "$productos.codigo_producto",
      cantidad_tickets_compra: { $addToSet: "$_id" }
    }
  },
  {
    $project: {
      _id: 0,
      codigo_producto: "$_id",
      cantidad_tickets_compra: { $size: "$cantidad_tickets_compra" }
    }
  },
  { $sort: { cantidad_tickets_compra: -1 } },
  { $limit: 5 }
]
```

6.4. Ejercicio 4 (NoSQL)

Respuesta

a) Respuesta

V Según el teorema CAP, es posible garantizar dos de tres características: particiones, consistencia y disponibilidad.

En una base de datos que no se partitiona, es posible garantizar (teóricamente) la consistencia y la disponibilidad.

b) **Respuesta**

F DynamoDB utiliza la consistencia eventual, que es la más laxa de las consistencias (secuencial, causal, eventual). Así, permite que las actualizaciones se propaguen a las réplicas de forma asíncrona, pudiendo tener un mismo ítem dos valores distintos en nodos diferentes y faltando temporalmente en la mantener consistencia.

c) **Respuesta**

V Como Cassandra sigue el concepto de *aggregate-oriented database* los datos de esa unidad de información , que en este caso son *wide rows*, se almacenan en un mismo nodo para facilitar el tráfico de consultas, entre otras cosas. Además, la clave que identifica a la *wide row* (clave de particionado) dice en qué nodo se va a guardar. No hay fragmentación. Cassandra asegura que toda la *wide row* va a estar en un mismo nodo, sin desperdigarla. Se la identifica mediante el hash de la clave de partición, resultado que indica dentro de qué nodo del anillo de hashing consistente está ese agregado. en la mantener consistencia.

d) **Respuesta**

F MongoDB también es una base de datos orientada al agregado, por lo que se pretende que la información de un documento esté alojada en un mismo nodo. Lo que permite Mongo es el *sharding*, que se basa en el particionamiento horizontal de las colecciones en *chunks*, que se distribuyen en nodos llamados *shards*, que contienen un subconjunto de documentos de cada colección. El *sharding* permite la distribución de una colección en distintos nodos, pero no de un documento entre distintos nodos.

e) **Respuesta**

V Según el teorema CAP, no se puede garantizar simultáneamente, en una base de datos distribuida, un máximo nivel de consistencia, disponibilidad y tolerancia a particiones. A lo sumo se pueden garantizar dos de las tres. Lo que usa en bases de datos distribuidas son las propiedades BASE

- *Basic availability*: el SGBD distribuido está siempre en funcionamiento, aunque eventualmente puede devolver un error o un valor desactualizado.
- *Soft state*: No es necesario que todos los nodos réplica guarden el mismo valor de un ítem en un determinado instante, por lo que no existe el “estado actual de la base de datos”.
- *Eventual consistency*: Si se dejan de producir actualizaciones, eventualmente todos los nodos alcanzarán el mismo estado.

6.5. Ejercicio 5 (Concurrencia y Transacciones)

a) **Respuesta**

F No es necesario que la transacción T_j comite para que T_i tenga el lock sobre el ítem X . Sólo es necesario que T_j lo libere respetando el protocolo, es decir: pidiendo todos los locks que necesitaba.

Nota: esto es considerando que el protocolo es 2PL. Si se estuviese usando el protocolo de 2PL estricto (S2PL), se tiene que *una transacción no puede adquirir un lock luego de haber liberado un lock que había adquirido, y además los locks de escritura sólo pueden ser liberados después de haber commitado la transacción*. Y el 2PL riguroso (S2PL) *no diferencia tipos de lock*. Los locks sólo pueden ser liberados después del commit. Por lo tanto, para S2PL y R2PL, esto sí es verdadero (excepto los locks de lectura en S2PL).

b) **Respuesta**

V En el 2PL no puede suceder $U(X) \dots L(X)$ para una misma transacción , puesto que se piden los locks de los ítems que se necesitan a medida que avanzan las operaciones atómicas de la transacción de manera incremental, y cuando no se van a lockear más ítems, se van *unlockeando*.

c) **Respuesta**

F El protocolo no impide la toma o liberación de locks sobre ítems distintos.

d) **Respuesta**

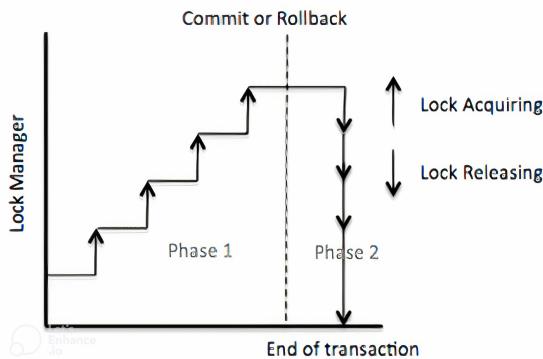
V Para realizar las operaciones atómicas, la transacción debe tener el lock del ítem sobre el cual va a operar. Sea éste $L_{SH}(X)$ ó $L(X)$.

e) **Respuesta**

V Misma justificación que d).

f) **Respuesta**

F La transacción T_i puede tener lock sobre ítems distintos porque sus operaciones atómicas así lo requieren. El comportamiento usual que se espera en este protocolo es



6.6. Ejercicio 6 (Concurrencia y Transacciones)

Respuesta

En *Snapshot Isolation* cada transacción guarda el estado de la base de datos y realiza sus operaciones atómicas como si sólo ella existiese, permitiendo así un mejor solapamiento. Otra ventaja es que no sucede la anomalía del fantasma, puesto que la transacción ve la misma snapshot a lo largo de su vida, mientras que en el control basado en locks se tiene que implementar el lock basado en predicados.

Como contraparte, una gran desventaja es que ocupa mucho espacio en memoria/disco al tener que mantener múltiples copias de los mismos ítems; y cuando hay conflictos del tipo escritura-escritura (W_{T_i}, W_{T_j}), hay que deshacer una de ellas.

7. Coloquio 2023-07-12

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 12 de julio de 2023

TEMA 20231C2				
CRT		Proc.	DR	
NoSQL		Rec.	CyT	
Nota: <input type="checkbox"/> Aprobado <input type="checkbox"/> Insuficiente				
Padrón: _____ Apellido: _____ Nombre: _____ Cantidad de hojas: _____				

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. (*Cálculo Relacional de Tuplas*) Se dispone de las siguientes 4 tablas con información sobre las asignaturas que cada profesor enseña en una universidad, y los departamentos a los que las mismas pertenecen:

- Departamentos(cod_dept, nombre_dept)
- Asignaturas(cod_dept, cod_interno_asignatura, nombre_asignatura, créditos)
- Profesores(legajo, apellido, nombre, fecha Alta)
- Enseña(legajo, cod_dept, cod_interno_asignatura)

Considere la siguiente consulta en *Cálculo Relacional de Tuplas* sobre los esquemas anteriores:
 $\{p \mid p.apellido, p.nombre \mid Profesores(p) \wedge (\exists e)(Enseña(e) \wedge e.legajo = p.legajo) \wedge (\exists d)(Departamentos(d) \wedge d.cod_dept = e.cod_dept \wedge d.nombre_dept = 'Matemática')$

Se pide:

- Explique en lenguaje coloquial qué realiza esta consulta.
- Traduzca la consulta al lenguaje SQL.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

3

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

4

- d) (*Bonus track*) Si sólo dispusiera de 500 bloques de memoria, ¿se le ocurre cómo podría llevar adelante la junta hash GRACE?
3. (*Diseño Relacional*) Considera una relación $R(A, B, C, D, E)$ con el siguiente conjunto de dependencias funcionales $F = \{B \rightarrow C, AC \rightarrow D, CE \rightarrow D, E \rightarrow A\}$. Indique si las siguientes afirmaciones son verdaderas (V) ó falsas (F), justificando su respuesta.

- $\{EC\}$ es clave candidata de la relación.
- F es un cubrimiento minimal.
- R se encuentra en Tercera Forma Normal (3FN).
- $(C \rightarrow D) \in F^+$

4. (*NoSQL*) Una red social almacena los cambios en las relaciones seguidor-seguido en una base de datos MongoDB. El siguiente documento JSON ejemplifica la estructura almacenada:

```

1  {
2    "_id": ObjectId("563c42d26b69693c4e61f98d"),
3    "follower_id": 21925,
4    "followed_id": 91930,
5    "modified_at": Date("2023-04-25 12:38:33"),
6    "change": 1
7  }
  
```

En particular, `follower_id` representa al usuario seguidor (llámémoslo A), `followed_id` representa al usuario seguido (B), `modified_at` representa la fecha en que se produjo el cambio en el vínculo, y `change` representa el sentido en que cambió (1 cuando A pasa a seguir a B, y -1 cuando A deja de seguir a B). Un usuario puede seguir/dejar de seguir a otro repetidas veces.

Escriba una consulta en MongoDB que encuentre para el usuario 91930 la cantidad de usuarios que lo siguen al día de hoy.

Para resolver esta consulta, puede utilizar el operador `$top`, que permite ordenar los resultados dentro de un grupo y cuya documentación se adjunta. Alternativamente, puede considerar que un usuario sigue a otro al día de la fecha cuando la suma de todos los `change` entre ellos da 1, como alternativa menos robusta pero satisfactoria para la solución.

5. (*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);
02 (BEGIN, T2);
03 (WRITE T1, X, 18);
04 (WRITE T2, Z, 6);
05 (COMMIT, T1);
06 (BEGIN CKPT, T2);
07 (BEGIN, T3);
08 (WRITE T3, Y, 4);
09 (WRITE T2, Z, 8);
10 (COMMIT, T2);
11 (END CKPT);
12 (WRITE T3, Z, 8);
13 (BEGIN, T4);
14 (WRITE T4, Y, 4);
  
```

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 realizarse en disco y en el log.

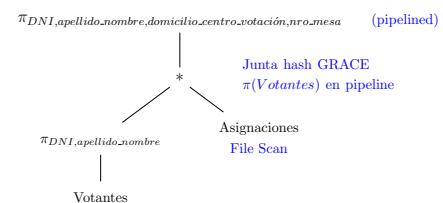
2. (*Procesamiento de Consultas*) La Junta Nacional Electoral se encuentra preparando los pabellones para las próximas elecciones, y a fines de lograr mayor transparencia tercerizó el trabajo de distribuir los votantes en los centros de votación. La empresa escogida recibió para cada votante un hash que lo identifica, e información aproximada sobre su domicilio. Con esta información, construyó una tabla de asignación de votantes con la siguiente estructura:

- Asignaciones(hash_votante, domicilio_centro_votación, nro_mesa)

Para construir el padrón definitivo, la Junta Nacional Electoral debe ahora combinar esta información con la verdadera identificación de cada votante, que se encuentra en la siguiente tabla:

- Votantes(DNI, apellido_nombre, hash_votante, domicilio, código_postal)

El padrón definitivo deberá contener únicamente el DNI del votante junto con su apellido y nombre y los datos del lugar de votación asignado. Para armarlo, se prepara el siguiente plan de ejecución:



Se pide:

- a) Estime el costo de este plan de ejecución en términos de cantidad de accesos a bloques de disco, asumiendo que se cuenta con $M=1500$ bloques de memoria disponibles. Considere para sus cálculos la siguiente información de catálogo:

VOTANTES	ASIGNACIONES
$n(\text{Votantes}) = 30.000.000$	$n(\text{Asignaciones}) = 30.000.000$
$B(\text{Votantes}) = 2.000.000$	$B(\text{Asignaciones}) = 2.000.000$
Tamaño medio de los campos:	
DNI: 4 bytes	
apellido_nombre: 21 bytes	
hash_votante: 6 bytes	
domicilio_votante: 15 bytes	
código_postal: 4 bytes	

- b) Indique qué cantidad k de particiones crearía para la junta hash GRACE, a efectos de poder llevarla a cabo correctamente. Justifique su respuesta.
- c) Indique cuál es el atributo por el que se deberá hashear cada tabla.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

4

6. (*Concurrencia y transacciones*) Considere el siguiente solapamiento de transacciones en un SGBD que emplea locks:

Transacción T ₁	Transacción T ₂	Transacción T ₃
begin	begin	begin
		lock(Z) leer.item(Z)
	lock(Y) leer.item(Y)	lock(X) leer.item(X)
		unlock(Z) commit
lock(X) leer.item(X) escribir.item(X)	lock(Z) leer.item(Z)	escribir.item(Z)
unlock(X)		lock(X) leer.item(X) escribir.item(Y) unlock(X) unlock(Y) unlock(Z) commit
		commit

Responda los siguientes ítems, justificando su respuesta.

- a) Indique si la ejecución respeta el *Protocolo de lock de 2 fases (2PL)*.

- b) Indique si este solapamiento es serializable.

- c) Indique si este solapamiento evita rollbacks en cascada.

\$top (aggregation accumulator)

Definition

\$top

New in version 5.2.

Returns the top element within a group according to the specified sort order.

Syntax

```
{  
  $top:  
  {  
    sortBy: { <field1>; <sort order>; <field2>; <sort order> ... },  
    output: <expression>  
  }  
}
```

Field	Necessity	Description
sortBy	Required	Specifies the order of results, with syntax similar to <code>\$sort</code> .
output	Required	Represents the output for each element in the group and can be any expression.

Example

Consider a gamescores collection with the following documents:

```
db.gamescores.insertMany([  
  { playerId: "PlayerA", gameId: "G1", score: 31 },  
  { playerId: "PlayerB", gameId: "G1", score: 33 },  
  { playerId: "PlayerC", gameId: "G1", score: 99 },  
  { playerId: "PlayerD", gameId: "G1", score: 1 },  
  { playerId: "PlayerA", gameId: "G2", score: 10 },  
  { playerId: "PlayerB", gameId: "G2", score: 14 },  
  { playerId: "PlayerC", gameId: "G2", score: 66 },  
  { playerId: "PlayerD", gameId: "G2", score: 80 }  
])
```

You can use the `$top` accumulator to find the top score in a single game.

```
db.gamescores.aggregate([  
  {  
    $match : { gameId : "G1" }  
  },  
  {  
    $group:  
    {  
      _id: "$gameId",  
      playerId:  
      {  
        $top:  
        {  
          output: [ "$playerId", "$score" ],  
          sortBy: { "score": -1 }  
        }  
      }  
    }  
])
```

The example pipeline:

- Uses `$match` to filter the results on a single gameId. In this case, G1.
- Uses `$group` to group the results by gameId. In this case, G1.
- Specifies the fields that are output for `$top` with `output : ["$playerId", "$score"]`.
- Uses `sortBy: { "score": -1 }` to sort the scores in descending order.
- Uses `$top` to return the top score in the game.

The operation returns the following results:

```
[ { _id: 'G1', playerId: [ 'PlayerC', 99 ] } ]
```

7.1. Ejercicio 1 (CRT)

Respuesta

La consulta se podría pensar como "Quiero a los profesores que no enseñen materias del departamento de matemática". En SQL, la consulta se traduciría a

```
SELECT p.legajo, p.apellido, p.nombre
FROM Profesores p
WHERE NOT EXISTS ( SELECT 1
                    FROM Enseña e
                    WHERE e.legajo = p.legajo
                      AND EXISTS ( SELECT 1
                                    FROM Departamentos d
                                    WHERE p.nro_socio = d.cod_dept
                                      AND d.nombre_dept = 'Matemática' ))
```

7.2. Ejercicio 2 (Procesamiento de Consultas)

Respuesta

En la tabla de **Votantes**, cada fila ocupa 50 bytes. A razón de 50 bytes por fila, hay 30,000,000 de filas repartidas en 2,000,000 bloques.

1. entonces la proyección de **(DNI, apellido_nombre)** reduce la cantidad de bloques a la mitad.

$$C_1 = B(1) = 1,000,000$$

2. Para calcular el costo de la junta es importante considerar que al estar en pipeline la proyección de la etapa previa no se considera el costo de lectura de sus bloques sino sólo del armado en memoria de las particiones. Entonces:

$$C_2 = 2 * B(1) + 3 * B(Asignaciones) = 2 * 1,000,000 + 3 * 2,000,000 = 8,000,000$$

3. La proyección se hace en pipeline y por atributos que conforman una clave del resultado de la junta, por lo que no sería necesario eliminar duplicados. Entonces:

$$C_3 = 0$$

Luego, el costo total es:

$$C_T = C_1 + C_2 + C_3$$

$$C_T = 9,000,000$$

Para definir la cantidad de particiones necesarias k para llevar a cabo la junta HASH GRACE es importante considerar las siguientes limitaciones:

1. $k \leq V(DNI, 1) = 30,000,000$, $k \leq V(hash_votante, Asignaciones) = 30,000,000$: no tendría sentido generar más particiones que la cantidad de valores distintos que puede tomar el atributo de junta, abarcarián la misma información. En este caso además
2. $k \leq M$: En la primera etapa hay que tomar al menos un bloque para armar cada partición.
3. $\frac{B(1)}{k}, \frac{B(Asignaciones)}{k} \leq M$: El tamaño de cada partición no debe superar la cantidad de memoria disponible para que en la segunda etapa se pueda realizar la junta de cada particiones (R_i, S_i) .

Dicho esto, un valor aceptable es $k = \sqrt{\max(B(R), B(S))} = \sqrt{B(Asignaciones)} = 1415$.

Finalmente, el atributo por el que se debe hashear cada tabla **SIEMPRE** es el atributo de junta.

7.3. Ejercicio 3 (Diseño Relacional)

Respuesta

- a) **Respuesta**

F Para ver si $\{EC\}$ es clave candidata hay que ver la clausura $\{EC^+\}$ cumple dos condiciones:

- $X \rightarrow R$ (el/los atributo/s implican a todos los de la relación R)
- No existe ningún Z en X tal que $Z \rightarrow R$ (condición de minimalidad. Quiere decir que ningún subconjunto de X cumple que implica toda la relación R)

$$EC^+ = \{E, C, A, D\}$$

Al usar las dependencias funcionales partiendo de $\{EC\}$ no se cubre todo el conjunto de atributos en R , por lo que $\{EC\}$ no es clave candidata.

b) **Respuesta**

V Para que un cubrimiento sea minimal se debe cumplir que:

- Las DF's tengan forma canónica: Todo implicado (lado derecho, LD) de una dependencia funcional DF de F tiene un único atributo, es decir es simple.

- $B \rightarrow C$ ✓
- $AC \rightarrow D$ ✓
- $CE \rightarrow D$ ✓
- $E \rightarrow A$ ✓

- Todo determinante (lado izquierdo, LI) de una dependencia funcional DF de F es reducido, en el sentido de no contener atributos redundantes o innecesarios.

Para verificar esta redundancia, se puede verificar si la clausura de los atributos del determinante incluye al implicado. En este caso, los posibles determinantes redundantes son: $\{AC, CE\}$.

- $A^+ = A \Rightarrow$ no incluye a su implicado (D) y por lo tanto no es redundante ✓
- $C^+ = C \Rightarrow$ no incluye a su implicado (D) y por lo tanto no es redundante ✓
- $C^+ = C \Rightarrow$ no incluye a su implicado (D) y por lo tanto no es redundante ✓
- $E^+ = E, A \Rightarrow$ no incluye a su implicado (D) y por lo tanto no es redundante. ✓

Como ninguna de las clausuras de los atributos individuales incluyen a su implicado, ninguno es redundante \Rightarrow cumple.

- F no contiene DF redundantes

Verificar esto implica analizar para cada DF $X \rightarrow Y$ si $Y \subset X_{F-\{X \rightarrow Y\}}^+$.

- $B \rightarrow C \Rightarrow$ $C \in B_{F-\{B \rightarrow C\}}^+$? No \Rightarrow No es redundante ✓
- $AC \rightarrow D \Rightarrow$ $D \in AC_{F-\{AC \rightarrow D\}}^+$? No \Rightarrow No es redundante ✓
- $CE \rightarrow D \Rightarrow$ $D \in CE_{F-\{CE \rightarrow D\}}^+$? No \Rightarrow No es redundante ✓
- $E \rightarrow A \Rightarrow$ $A \in E_{F-\{E \rightarrow A\}}^+$? No \Rightarrow No es redundante ✓

Luego, el cubrimiento es minimal.

c) **Respuesta**

F Decimos que una relación está en tercera forma normal (3FN) cuando no existen dependencias transitivas $CK_i \rightarrow Y$ de atributos no primos (i.e. $Y \not\subseteq \bigcup_i CK_i$), con CK_i clave candidata.

Una definición equivalente es que para toda dependencia funcional no trivial $X \rightarrow Y$, o bien X es superclave, o bien $Y - X$ contiene sólo atributos primos.

En esta relación R con las dependencias funcionales F no se cumple, para la clave $PK=\{E, B\}$ que en particular es la única candidata, que para la dependencia funcional no trivial $AC \rightarrow D$ el determinante sea superclave y por lo tanto no está en 3FN.

d) **Respuesta**

F Dado un conjunto F de DF's, sea la clausura F^+ el conjunto de todas las DF's que pueden inferirse usando los Axiomas de Armstrong.

Dicho esto, usando estos axiomas no se puede llegar a $C \rightarrow D$, y por lo tanto es falso.

7.4. Ejercicio 4 (NoSQL)

Respuesta

```
[  
  {  
    $match: {"followed_id": 91930, "change": 1}  
  },  
  {  
    $group: {_id: {"follower_id": "$follower_id", "followed_id": "$followed_id"},  
    estado_seguimiento:  
      {  
        $top:  
          {  
            output: "$change",  
            sortBy: { "change": -1 }  
          }  
      }  
    },  
    {$count: "estado_seguimiento"  
  }  
]
```

7.5. Ejercicio 5 (Recuperación)

Respuesta

Como lo primero que encuentra el algoritmo al recorrer hacia atrás el *log* es un END CKPT entonces sólo deberá retroceder hasta su BEGIN CKPT .

Luego vemos que T_2 commiteó pero T_3 y T_4 no, por lo que hay que deshacer las instrucciones de T_3 y T_4 en disco, escribiendo:

1. Y=4 por T_4 línea 14
2. Z=8 por T_3 línea 12
3. X=18 por T_3 línea 08

Finalmente se escribe en el log las instrucciones (ABORT, T_3) y (ABORT, T_4) y se vuelca a disco.

7.6. Ejercicio 6 (Concurrencia y Transacciones)

a) **Respuesta**

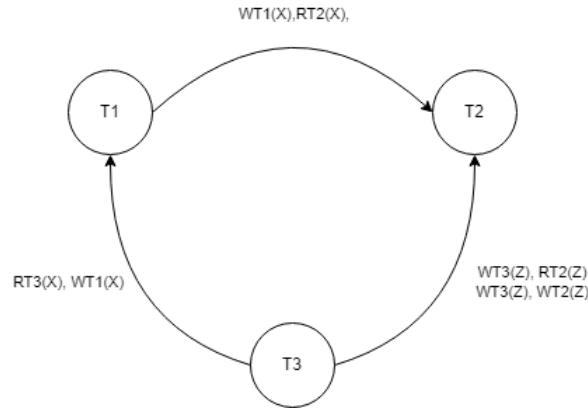
La ejecución presentada respeta el protocolo de lock de 2 fases (2PL). Viendo la tabla del enunciado, se puede ver que en cada transacción se van adquiriendo los locks a medida que se van necesitando los ítems y luego de haber adquirido todos los necesarios se hace el unlock de esos ítems.

b) **Respuesta**

Viendo el solapamiento

$b_{T_1}, b_{T_2}, b_{T_3}, L_{T_3}(Z), R_{T_3}(Z), L_{T_2}(Y), R_{T_2}(Y), L_{T_3}(X), R_{T_3}(X), W_{T_3}(Z), U_{T_3}(Z), L_{T_2}(Z), R_{T_2}(Z), U_{T_3}(X), c_{T_3}$
 $L_{T_1}(X), R_{T_1}(X), W_{T_1}(X), W_{T_2}(Z), U_{T_1}(X), L_{T_2}(X), R_{T_2}(X), W_{T_2}(Y), U_{T_2}(X), U_{T_2}(Y), U_{T_2}(Z), c_{T_2}, c_{T_1}$

Y el grafo de precedencias



Se puede ver que el grafo es acíclico, por lo que es serializable el solapamiento.

- c) Para evitar los rollbacks en cascada es necesario que una transacción no lea valores que aún no fueron commiteados. Esto es más fuerte que la condición de recuperabilidad. Esta definición implica que quedan prohibidos los conflictos de la forma $(W_{T_i}(X), R_{T_j}(X))$ sin que en el medio exista un commit c_{T_i} , cosa que no se cumple en este caso. Particularmente en $W_{T_3}(Z), U_{T_3}(Z), L_{T_2}(Z), R_{T_2}(Z), U_{T_3}(X), c_{T_3}$, donde T_2 está leyendo el ítem Z que fue modificado por T_3 pero todavía no hizo el commit.

8. Coloquio 2023-07-05

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 5 de julio de 2023

TEMA 20231C1					Padrón: _____
Mod.	Proc.	CRT			Apellido: _____
SQL	NoSQL	NoSQL			Nombre: _____
Nota:					Cantidad de hojas: _____
					<input type="checkbox"/> Aprobado <input type="checkbox"/> 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. (*Modelado*) Una nueva plataforma de streaming de música online quiere diseñar una base de datos para responder a los requisitos de su modelo de negocio. Dibuja un modelo entidad interrelación que cumpla con los siguientes requisitos:

- Los usuarios de la plataforma se identificarán con un nombre de usuario, y se almacenará también su fecha de nacimiento y su número de tarjeta de crédito.
- El elemento básico de reproducción de la plataforma es la “canción”. Una canción se identifica con un id interno del sistema, posee un nombre, y es interpretada por un artista. Un artista puede ser un músico individual, o bien puede ser un grupo compuesto por varios músicos.
- Los discos son conjuntos de canciones, no necesariamente cantadas por un mismo artista.
- Los usuarios pueden crear listas de reproducción, que son conjuntos de canciones. Las distintas listas de un mismo usuario se identifican a partir del nombre, aunque usuarios distintos pueden tener listas con igual nombre. Una lista de reproducción puede ser pública o privada.
- Cuando la lista de reproducción es pública, otros usuarios pueden seguirla.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires) 3

4. (*SQL*) Una empresa de gestión de backups como servicio almacena versiones encriptadas de los datos de sus clientes en data centers ubicados en distintas ciudades del mundo. El estándar de seguridad manejado por la empresa exige que todo archivo disponga de copias de seguridad actualizadas en al menos 2 ciudades distintas.

Adicionalmente, la empresa mantiene una base de datos indicando en qué discos de qué data centers posee copias de qué archivos:

- Archivos(file_id, client_id, nombre, fecha_último_backup)
- CopiasSeguridad(file_id, disk_id, fecha_backup)
- Discos(disk_id, marca, capacidad, data_center_id)
- DataCenters(data_center_id, ciudad, latitud, longitud)

Escriba una consulta en SQL que encuentre los id's de los archivos que no tienen copias de seguridad actualizadas en al menos dos ciudades distintas.

5. (*NoSQL*) Son escasas pero existentes las situaciones en que dos personas hermanas se casan con personas que también son hermanas entre sí. Un ejemplo de esta situación se dió con los hermanos Rogelio y Ramiro Funes Mori, quienes se casaron con Jorgelina y Rocío Díaz, hermanas entre sí. Para investigar la frecuencia con que se dan estas situaciones en la Argentina, accedemos a una base de datos del Registro Nacional de las Personas almacenada en Neo4j, con la siguiente estructura:

Cada persona con DNI argentino se encuentra registrada como un nodo de tipo PERSONA:

```
1 (p1: PERSONA { DNI: 42109385; apellido: 'Funes Mori', nombre: 'Rogelio' })
2 (p2: PERSONA { DNI: 41378559; apellido: 'Diaz', nombre: 'Jorgelina' })
```

La filiación de las personas queda registrada a través de una interrelación de tipo HIJO_DE:

```
1 MATCH (p1: PERSONA { DNI: 42109385 } ), //Conectamos a Rogelio con su padre
2   (p2: PERSONA { DNI: 16518070 } )
3 CREATE (p2)-[:HIJO_DE]-(p1);
```

Por último, el matrimonio entre dos personas queda registrado como una interrelación CASADA_CON:

```
1 MATCH (p1: PERSONA { DNI: 42109385 } ), //Casamos a Jorgelina con Rogelio
2   (p2: PERSONA { DNI: 41378559 } )
3 CREATE (p2)-[:CASADA_CON]-(p1);
```

Encuentre en esta base de datos las situaciones en que dos personas hermanas se encuentran casadas con dos personas hermanas, escribiendo una consulta en Cypher que devuelva en cada línea los nombres de los integrantes de cada una de las dos parejas.

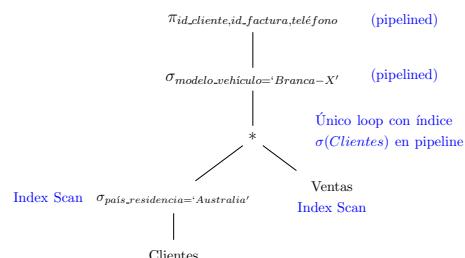
6. (*NoSQL II*) Explique en qué consiste el concepto de *agregado* descripto por Martin Fowler y proveniente del *Domain Driven Design*. Indique qué relevancia tiene en el contexto de las bases de datos NoSQL, y ejemplifique de qué forma es implementado en algún tipo de ellas.

2. (*Procesamiento de Consultas*) El fabricante de vehículos inteligentes Elton Rust lanzó hace unos meses su último vehículo Branca-X, con funcionalidades de punta como parabrisas con realidad aumentada, selección automática de ruta y deliveries gratuitas de comida cada 500km en ruta. Desafortunadamente, un error de concurrencia en el software del vehículo hace que el dron del delivery impacte contra el vehículo antes de que el mismo abra la ventanilla. En Australia, por cuestiones legales, Elton necesita retirar del mercado a todos los vehículos del modelo Branca-X. Para encontrar a los flamantes propietarios de los mismos, Elton dispone de las siguientes 3 tablas:

- Vehículos(modelo, tipo_vehículo, fecha_lanzamiento)
- Ventas(id_factura, id_cliente, modelo_vehículo, precio_venta)
- Clientes(id_cliente, apellido, nombre, país_residencia, teléfono)

Además, cuenta con un índice de clustering para la tabla *Clientes* por el atributo *país_residencia*, mientras que la tabla *Ventas* cuenta con un índice secundario sobre *id_cliente*.

Para buscar a los propietarios de vehículos Branca-X en Australia, Elton preparó el siguiente plan de ejecución:



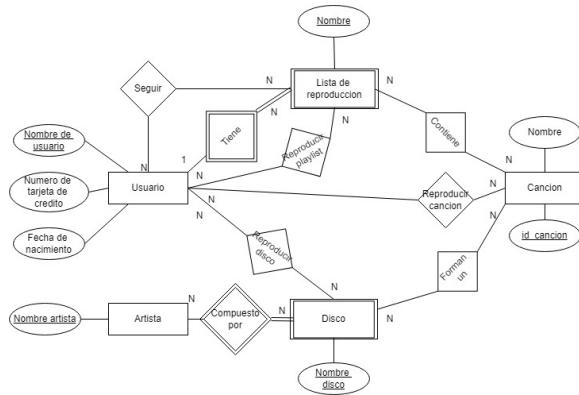
Estime el costo del plan de ejecución en términos de cantidad de accesos a bloques de disco, suponiendo que dispone de una cantidad mínima de memoria. Considere para el cálculo la siguiente información de catálogo:

CLIENTES	VENTAS
n(Clientes) = 1.000.000	n(Ventas) = 4.000.000
B(Clientes) = 100.000	B(Ventas) = 400.000
V(país_residencia, Clientes) = 100	V(id_cliente, Ventas) = 1.000.000
V(modelo_vehículo, Ventas) = 80	

3. (*Cálculo Relacional de Tuplas*) Para las mismas tablas del Ejercicio 2, encuentre los nombres de los países en que residen clientes que compraron algún vehículo modelo Branca-X.

8.1. Ejercicio 1 (Modelado)

Respuesta



8.2. Ejercicio 2 (Procesamiento de Consultas)

Respuesta

Suponiendo una cantidad mínima de memoria ($M = 2$), el costo de cada etapa sería:

1. En la selección supongo $H(I(pais_residencia, Clientes)) = 4$

$$C_1 = H(I(pais_residencia, Clientes)) + \lceil \frac{B(Clientes)}{V(pais_residencia, Clientes)} \rceil = 4 + \lceil \frac{100,000}{100} \rceil = 1004$$

Y de esta etapa se toman $n(1) = \frac{n(Clientes)}{V(pais_residencia='Australia', Clientes)} = 10,000$ filas que corresponden a `pais_residencia='Australia'` que caben en $B(1) = 1000$ bloques, y se los toma como la cantidad de accesos a bloques de disco de esta etapa.

2. Para calcular el costo de la junta es importante considerar que al estar en pipeline la selección de la etapa previa no se considera el costo de lectura de sus bloques sino sólo la búsqueda en memoria de los bloques de `Ventas`. Además se considera una altura del índice secundario sobre `id_cliente` en `Ventas` de 4. Entonces:

$$C_2 = n(1) * (H(I(id_cliente, Ventas) + \lceil \frac{n(Ventas)}{V(id_cliente, Clientes)} \rceil) = 10,000 * (4 + \lceil \frac{4,000,000}{1,000,000} \rceil) = 80,000$$

Como por cada fila en pipeline de la etapa previa se debe acceder a disco para buscar las que coincidan en el atributo de junta `id_cliente` en la tabla `Ventas` en disco, en total se acceden a unas $n(2) = 80,000$ filas, que se corresponden con $B(2) = 8,000$ bloques en disco.

3. Como la selección y proyección finales se hacen en pipeline, no agregan costo en términos de accesos a bloques de disco.

$$B_3 = 0$$

Luego, el costo total es:

$$C_T = B_1 + B_2 + B_3$$

$$C_T = 81,000$$

8.3. Ejercicio 3 (CRT)

Respuesta

$$\begin{aligned} \{c.pais_residencia | & \quad Clientes(c) \wedge \\ & (\exists s)(Ventas(s)(s.id_cliente = c.id_cliente \wedge \\ & (\exists v)(Vehiculos(v) \wedge s.modelo_vehiculo = v.modelo \wedge \\ & modelo = 'Branca - X'))) \} \end{aligned}$$

8.4. Ejercicio 4 (SQL)

Respuesta

```
SELECT a.file_id
FROM Archivos
WHERE a.file_id NOT EXISTS (
    SELECT c1.file_id
    FROM CopiasSeguridad c1
        INNER JOIN Discos d1 USING (disk)
        INNER JOIN DataCenters dc1 USING (data_center_id)
    WHERE EXISTS (
        SELECT *
        FROM CopiasSeguridad c2
            INNER JOIN Discos d2 USING (disk)
            Discos JOIN DataCenters dc2 USING (data_center_id)
            WHERE (c1.file_id = c2-file_id) AND (dc1.ciudad <> dc2.ciudad)
    )
)
```

8.5. Ejercicio 5 (NoSQL)

Respuesta

```
MATCH (hermano1:Persona -[:HIJO_DE]->padre1:Persona<-[:HIJO_DE]-hermano2:Persona),
(hermano2-[:CASADA_CON]->pareja2:Persona),
(pareja2-[:HIJO_DE]->padre2:Persona<-[:HIJO_DE]-pareja1:Persona),
(hermano1-[:CASADA_CON]->pareja1)
WHERE hermano1.DNI>hermano2.DNI
RETURN DISTINCT hermano1,pareja1, hermano2, pareja2
```

8.6. Ejercicio 6 (NoSQL)

Respuesta

Martin Fowler describe al agregado como la unidad natural de información a guardar en la base de datos: en una base orientada a documentos, el agregado es un documento; en una base *wide column* cada columna representa un agregado. Al guardar cada agregado en la base de datos como una unidad se logra un mejor flujo de datos a la hora de realizar las consultas, entendiéndose como la extracción de datos de la base (*data retrieval*). Algunos ejemplos son MongoDB, donde el agregado es un documento (generalmente JSON); Cassandra, donde el agregado es la columna; o como Dynamo, donde el agregado es el valor.

9. Coloquio 2023-03-01

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 1 de marzo de 2023

TEMA 20222C5				
CRT	Proc.	SQL		
NoSQL	NoSQL	CyT		
Nota:				
<input type="checkbox"/> Aprobado <input type="checkbox"/> 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. (*CRT*) En una plataforma de fútbol online, cada jugador está caracterizado por una medición de las siguientes habilidades: velocidad, defensa, pase, disparo, resistencia. Esta información, junto con el equipo al que pertenece el jugador, se encuentra relevada en la siguiente tabla:

- Jugadores(nombre_jugador, nombre_equipo, velocidad, defensa, pase, disparo, resistencia)
(‘Angel Correa’, ‘Atlético de Madrid’, 89, 63, 90, 93, 85)

Escriba una consulta en *Cálculo Relacional de Tuplas* que encuentre los nombres de los jugadores que superan en todas las habilidades a todos los demás jugadores de su mismo equipo.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

3

4. (*NoSQL*) Indique si las siguientes afirmaciones sobre *Dynamo* son verdaderas o falsas, justificando su respuesta.

- En el mecanismo de hashing consistente no sólo las claves deben ser hasheadas, sino también los nodos.
- En Dynamo, si un nodo se cae temporalmente, las consultas sobre claves ese nodo almacenará no podrán ser respondidas por la base de datos hasta tanto ese nodo no vuelva a levantarse.
- El uso de Merkle trees les permite a los nodos de Dynamo resolver los problemas de inconsistencias en sus datos.
- El mecanismo de hashing consistente es lo que le permite a Dynamo garantizar un nivel de consistencia eventual.

5. (*NoSQL*) Teobaldo es apasionado de la historia, y le gusta analizar datos sobre los reinados de la Edad Media. Para ello construyó una base en Neo4j con información sobre los miembros de las distintas familias reales, sus filiaciones, matrimonios, y reinados.

En esta base, los nodos representan a las personas y a los territorios, como muestra el siguiente ejemplo:

```
1 CREATE (p1:Persona {nombre: 'Alfonso III de Aragón', año_nacimiento: 1265,
2                     ciudad_nacimiento: 'Valencia', año_fallecimiento: 1291,
3                     ciudad_fallecimiento: 'Barcelona'});
4 CREATE (t1:Territorio {nombre: 'Reino de Aragón', año_formación: 1035,
5                     año_disolución: 1707});
```

Asimismo, las aristas de este grafo representan interrelaciones de tipo SE_CASÓ_CON, HIJO_DE y REINÓ, como se ilustra a continuación:

```
1 MATCH (p1: Persona {nombre: 'Alfonso III de Aragón'}),
2       (p2: Persona {nombre: 'Léonor de Inglaterra'}),
3       (p3: Persona {nombre: 'Enrique II'});
4 CREATE (p1)-[:SE_CASÓ_CON]->(p2),
5       (p2)-[:HIJO_DE]->(p3);
6
7 MATCH (p1: Persona {nombre: 'Alfonso III de Aragón'}),
8       (t1: Territorio {nombre: 'Reino de Aragón'});
9 CREATE (p1)-[:REINÓ {inicio_reinado: 1285, fin_reinado: 1291}]->(t1);
```

Teobaldo quisiera identificar las situaciones en que un hijo/a de un/a rey/reina de Inglaterra, sin llegar nunca a ser rey/reina de Inglaterra, se casó con un rey/reina de Francia.

Escriba una consulta en *Cypher* que encuentre los nombres de estas personas que, siendo hijas de alguien que reinó en el Reino de Inglaterra pero sin nunca haberlo reinado, se hayan casado con alguien que reinó en el Reino de Francia.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

2. (*Procesamiento de Consultas*) La FIFA quiere analizar la repercusión que tuvieron los premios *The Best* en distintos países del mundo. Para ello dispone de datos de la red social *Twitter* extraídos en la última semana. La información proveniente de esta red social se encuentra concentrada en las siguientes dos tablas:

- Users(user_id, nombre_usuario, ciudad, país)
- Usos_Hashhtags(tweet_id, hashtag, user_id)

La tabla *Users* cuenta además con un índice secundario por el atributo *país*, mientras que la tabla *Usos_Hashhtags* cuenta con un índice de clustering por el atributo *hashtag*, ambos de altura 4. Se dispone también de la siguiente información de catálogo:

USERS	USOS_HASHTAGS
n(Users) = 10.000.000	n(Usos_Hashhtags) = 5.000.000
B(Users) = 1.000.000	B(Usos_Hashhtags) = 1.000.000
V(país, Users) = 200	V(hashtag, Usos_Hashhtags) = 50.000
	V(user_id, Usos_Hashhtags) = 500.000

Adicionalmente, la tabla *Usos_Hashhtags* cuenta con un histograma para el atributo *hashtag*, y ese histograma indica que el hashtag #TheBest fue uno de los trending topics, con 500.000 usos en la base de datos.

Los analistas de la FIFA quisieran calcular la cantidad de veces que se utilizó el hashtag #TheBest en cada país del mundo, mostrando para cada país su nombre y la cantidad de usos del hashtag #TheBest.

Considerando que sólo dispone de M=1000 bloques de memoria, se pide:

- Diseñe un plan de ejecución eficiente para la consulta, que permita resolverla con un costo razonablemente bajo.
- Estime el costo del plan de ejecución propuesto.

3. (*SQL*) Emil Zatopek fue uno de los más grandes corredores de la historia del atletismo, habiendo conseguido el oro olímpico en los 5.000 y 10.000 metros llanos, y en la maratón. En las siguientes tablas, la *Federación Internacional de Atletismo* tiene registro de las marcas de los corredores en las distintas competiciones oficiales realizadas:

- Atletas(cod_atleta, nombre_apellido, fecha_nacimiento, nacionalidad)
- Competiciones(cod_competicion, tipo, ciudad, fecha)
- Marcas(cod_atleta, cod_competicion, marca)

Escriba una consulta en SQL que liste los códigos y nombres de aquellos atletas que alguna vez superaron a Emil Zatopek en una competición de tipo ‘10000 metros llanos’.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires) 4

6. (*Concurrencia y Transacciones*) Indique si las siguientes afirmaciones sobre concurrencia son verdaderas o falsas, justificando brevemente su respuesta.

- En el método de control de concurrencia *Snapshot Isolation* no es posible que ocurra la anomalía del fantasma.
- En el método de control de concurrencia basado en timestamps no pueden ocurrir deadlocks.
- Bajo *Snapshot Isolation* nunca es necesario abortar una transacción para garantizar serializabilidad.
- Bajo 2PL (*two-phase locking*) todo solapamiento de transacciones resulta recuperable.
- Bajo *Snapshot Isolation* nunca puede ocurrir una lectura sucia.

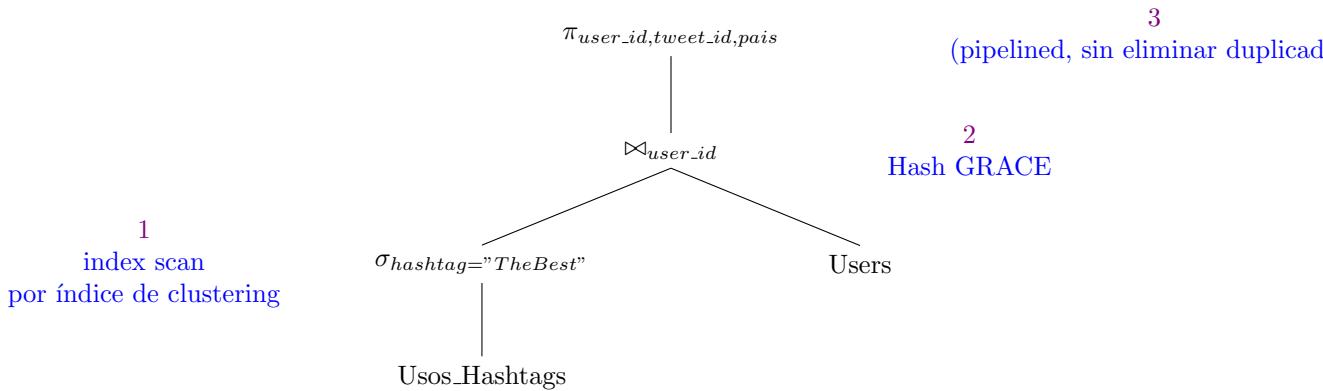
9.1. Ejercicio 1 (CRT)

Respuesta

$$\{j_1.nombre_jugador | \quad Jugadores(j_1) \wedge (\#j_2)(Jugadores(j_2) \wedge \\ j_1.nombre_equipo = j_2.nombre_equipo \wedge \\ j_1.velocidad \leq j_2.velocidad \wedge \\ j_1.pase \leq j_2.pase \wedge \\ j_1.resistencia \leq j_2.resistencia \wedge \\ j_1.disparo \leq j_2.disparo \wedge \\ j_1.defensa \leq j_2.defensa)\}$$

9.2. Ejercicio 2 (Procesamiento de Consultas)

Respuesta



- Para el costo de la instrucción 1 es necesario tener en cuenta que existe un índice de clustering para buscar por hashtag en la tabla de **Usos_HashTags**.

Entonces, sabiendo que hay que buscar los datos en disco

$$C_1 = H(I(hashtag, Usos_HashTags)) + \lceil \frac{B(Usos_HashTags)}{V(hashtag, Usos_HashTags)} \rceil$$

$$C_1 = 4 + \lceil \frac{1,000,000}{50,000} \rceil$$

$$C_1 = 24$$

Entonces C_1 es el costo de acceso a disco para la instrucción de selección.

El resultado son 500,000 filas, dadas por la información del histograma, que entran en 100,000 bloques de memoria (en cada bloque entran $F = 5 = \frac{n(Usos_HashTags)}{B(Usos_HashTags)}$).

- Para calcular el costo de 2 se puede asumir pipelining: a medida que las filas seleccionadas van apareciendo, se va procesando la junta. Además se usa la junta Hash GRACE. Como una de las tablas de \bowtie_{user_id} se procesa en pipeline, sólo se tendrá el costo de armar las particiones de cada tabla y de buscar la tabla **Users** en memoria. Éste costo de armado de particiones es de $2B(R)$ para cada tabla R , $\Rightarrow 2B(Users)$.

Luego es necesario considerar el costo de comparaciones entre particiones (R_i, S_i) , que si ambas están en disco sería $B(R) + B(S)$, pero como una de las tablas ya está en memoria sería únicamente $B(S)$, entonces el costo total sería:

$$C_2 = 2 * B(1) + 3B(Users)$$

$$C_2 = 2 * 100,000 + 3 * 1,000,000$$

$$C_2 = 3,200,000$$

La tabla **Users** no entra en memoria, por lo que la cantidad de particiones debe respetar $k < M, k < V(user_id, USers) = 1,000,000, k < V(user_id, Usos_HashTags) = n(Users) = 1,000,000$. Un valor lógico de la cantidad de particiones sería $k = 317$.

3. En el paso 3 no hay costo de acceso a disco, pues se procesan las tuplas en pipeline, a medida que termina su procesamiento en la etapa previa. Además no hay que eliminar duplicados por ser el atributo de proyección superclave.

Luego el costo total es

$$C_T = C_1 + C_2 + C_3$$
$$C_T = 24 + 3,200,000 = 3,200,024$$

9.3. Ejercicio 3 (SQL)

Respuesta

Se puede solucionar usando sólo una consulta con una subconsulta, pero es más cómodo y visualmente entendible usando la cláusula WITH. Como se estpa usando agregación en dicha cláusula, el resultado va a ser sólo una fila, y se elige una sola columna: la de marcas, con lo cual resulta un sólo elemento.

```
SELECT DISTINCT cod_atleta, nombre_apellido
FROM Atletas a INNER JOIN Marcas m USING(cod_atleta)
    INNER JOIN Competiciones c USING(cod_competicion)
WHERE tipo='10000 metros llanos' AND
    marca < (    SELECT MIN(marca)
                    FROM Atletas a INNER JOIN Marcas m USING (cod_atleta)
                        INNER JOIN Competiciones c USING(cod_competicion)
                            WHERE tipo='10000 metros llanos' AND
                                a.nombre_apellido= 'Emil Zatopek'
                );
```

9.4. Ejercicio 4 (NoSQL)

Respuesta

a) **Respuesta**

V El hashing de módulo N se hashean sólo las claves, pero en el hashing consistente se hashean ambas, reduciendo así la cantidad de movimientos de pares necesarios cuando cambia la cantidad de nodos. Esto hace que sea muy sencillo agregar nodos en forma dinámica, con un impacto mínimo.

b) **Respuesta**

F En Dynamo, al caerse un nodo se reestructura el anillo de hashing, moviendo sólo los pares que estaban en el nodo caído. De esta forma, los datos estarían temporalmente inhabilitados mientras se los mueven, pero no hasta que se vuelva a levantar el nodo caído.

c) **Respuesta**

V El uso de *Merkle trees* permite mantener sincronizadas las réplicas, y sortear así los problemas de inconsistencia en los datos. Como Dynamo funciona bajo el modelo de consistencia eventual, puede que tolere grados más altos de inconsistencia, teniendo valores de un mismo par (*key,value*) distintos para cada réplica. Los *Merkle trees* ayudan su sincronización .

d) **Respuesta**

F El hashing consistente reduce la cantidad de movimientos de pares necesarios cuando cambia la cantidad de nodos S, pero no ayuda a la sincronización de los datos a través de las réplicas.

9.5. Ejercicio 5 (NoSQL)

Respuesta

```
MATCH (h1:Persona)-[:HIJO_DE]->(p1:Persona)-[:REINO]->(ing:Territorio{'Reino de Inglaterra'}),  
      (h2:Persona)-[:REINO]->fra:Territorio{'Reino de Francia'},  
      h1-[:SE_CASO_CON]->h2  
WHERE NOT EXISTS (  
    MATCH (h1-[:REINO]->ing))
```

```
    RETURN h1
)
RETURN h1.nombre
```

9.6. Ejercicio 6 (Concurrencia y Transacciones)

a) **Respuesta**

Dado que cada transacción tiene una copia del estado de la base de datos del momento en el que inició y opera sobre ella, no es posible que suceda la anomalía del fantasma.

b) **Respuesta**

Al no usar locks, está exento de deadlocks.

c) **Respuesta**

En el caso de que hubiere conflictos de escritura-escritura entre dos transacciones que intentan manejar el mismo ítem, habría que abortar una de las transacciones.

d) **Respuesta**

En el caso de S2PL o R2PL sería verdadero, pues garantizan serializabilidad, recuperabilidad y evitan rollbacks en cascada. Sin embargo, dado que en 2PL no impone restricciones sobre el momento de commitear de las transacciones T_i , T_j es posible que se de la condición de no recuperabilidad. Ejemplo: ejercicio 6) coloquio 2023-07-12 c).

e) **Respuesta**

Dado que cada transacción tiene una copia del estado de la base de datos del momento en el que inició y opera sobre ella, no es posible que suceda la anomalía de lectura sucia.

10. Coloquio 2023-02-22

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 22 de febrero de 2023

TEMA 20222C4			Padrón: _____
SQL	Proc.	NoSQL	
NoSQL	CyT	Rec.	
Nota:			<input type="checkbox"/> Aprobado <input type="checkbox"/> 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*) El INDEC releva mensualmente los precios de distintos productos en el mercado, a fin de brindar una estimación del índice de inflación interanual. Todos los precios relevados se mantienen en las siguientes dos tablas:

- **Productos**(cod_barras_producto, nombre_producto, categoría)
(291478431208, 'Detergente Cof Crema', 'LIMPIEZA')
- **PreciosRelevados**(cod_barras_producto, fecha, precio)
(291478431208, '01-02-2023', 149.99)

En base a estas tablas, el INDEC necesita calcular la inflación interanual en el rubro LIMPIEZA a febrero de 2023, que se calcula como el promedio de los aumentos relativos en productos de limpieza entre las fechas 01-02-2022 y 01-02-2023. En otras palabras, se debe calcular el aumento relativo en cada producto de limpieza entre una fecha y la otra, y luego promediar todos los aumentos relativos. Escriba una consulta en SQL que estime la inflación interanual en el rubro LIMPIEZA a febrero de 2023.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

3

- (c) Si el sharding se realizó por un atributo A, y luego se hace una consulta por un valor específico de un atributo B, esa consulta será poco eficiente porque cada nodo deberá leer todo su base local en busca de documentos que cumplen con esa condición.
- (d) Cuando se hace una consulta por un valor específico del atributo de sharding, el resultado será un único documento, y el mismo podrá ser hallado en forma muy eficiente por el nodo que lo almacena.
4. (*NoSQL*) Indique si las siguientes afirmaciones sobre los LSM-trees (*log-structured merge trees*), son verdaderas o falsas, justificando brevemente su respuesta.
- Cuando se hace un rolling merge entre la estructura C_0 en memoria y la estructura C_1 en disco, los datos de C_0 no se van agregando al C_1 existente, sino que se guardan en una nueva estructura C_1^* en disco.
 - Cuando un usuario inserta un par (k, v) , el mismo se agrega en C_0 en memoria – asumiendo que haya espacio –, exista o no ya una entrada para k en C_1 en disco.
 - La estructura C_0 del LSM-tree es mantenida en memoria volátil. Por lo tanto, el gestor no la garantiza al usuario la durabilidad de los datos que se encuentran escritos en C_0 .
 - Si se buscan los datos asociados a una clave k en C_0 y no se encuentra ninguna entrada para k , entonces se deberá buscar en el nivel C_1 .
 - La estructura C_0 en memoria conviene implementarla con un B-tree.

5. (*Concurrencia y Transacciones*) Complete el siguiente cuadro referente un solapamiento *tentativo* de tres transacciones bajo el esquema de control de concurrencia basado en timestamps, indicando en cada fila qué cambios se producen en los `read_TS` y `write_TS` a partir de la operación respectiva. Detenga el análisis cuando encuentre que una transacción deberá ser abortada con el fin de garantizar la serializabilidad, indicando de qué transacción y operación se trata, y cuál es la regla que viola. Consideré que los timestamps de las transacciones coinciden con sus subíndices.

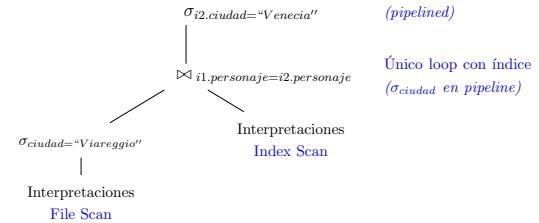
A modo de ejemplo se ha completado la primera fila, $R_{T_6}(Y)$.

	T_6	T_8	T_9	X	Y	Z
Valores inic.	TS=6	TS=8	TS=9	read_TS=0 write_TS=0	read_TS=0 write_TS=0	read_TS=0 write_TS=0
$R(Y)$	$W(X)$	$W(Z)$	$R(X)$		$read_TS(Y)=6$	
	$R(Z)$					

2. (*Procesamiento de Consultas*) En Italia, el carnaval se celebra en cientos de ciudades con la presencia de personajes típicos o "máscaras". Algunos de estos personajes típicos son Arlequín, Pantaleón o Colombina, por ejemplo. Luego de celebrado el Carnaval 2023, la *Red Italiana de Carnaval* quiere poner en contacto a personas que interpretaron a la misma máscara en distintas ciudades. Para comenzar con este proyecto, se pondrá en contacto a personas que interpretaron al mismo personaje en el carnaval de Venecia y en el carnaval de Viareggio. Para realizar esta tarea, se cuenta con una tabla en una base de datos relacional que registra los datos de las personas que tuvieron algún papel en algún carnaval de 2023:

- **Interpretaciones**(apellido.nombre, mail, ciudad, personaje)
(‘Elsa Rivetti’, ‘erivetti@gmail.com’, ‘Venecia’, ‘Arlequín’)

Para obtener el resultado requerido, se utilizará el siguiente plan de ejecución:



La tabla dispone de un índice de clustering de tipo árbol por **personaje**, de altura 3. Además, puede considerar para sus cálculos la siguiente información de catálogo:

INTERPRETACIONES
n(Interpretaciones) = 20.000
B(Interpretaciones) = 2.000
V(ciudad, Interpretaciones) = 20
V(personaje, Interpretaciones) = 100

Se pide:

- Estime el costo del plan de ejecución diseñado.
- Estime el tamaño del resultado, en términos de cantidad de tuplas.

3. (*NoSQL*) Para una colección en *MongoDB* bajo un esquema de *sharding* con replicación, indique si las siguientes afirmaciones son verdaderas o falsas, justificando su respuesta.

- Cuando se buscan documentos con un valor específico de un atributo A, se deberá consultar a todos los replica sets, que deberán buscar en su base local.
- Cuando se inserta un nuevo documento en un replica set, se deberá esperar a que el documento se escriba en todos los nodos de ese replica set antes de devolverle el control al usuario.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires) 4

6. (*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);
02 (BEGIN, T2);
03 (WRITE T1, A, 13);
04 (WRITE T2, C, 7);
05 (COMMIT, T1);
06 (BEGIN CKPT, T2);
07 (BEGIN, T3);
08 (WRITE T3, A, 8);
09 (WRITE T2, B, 15);
10 (COMMIT, T2);
11 (END CKPT);
12 (BEGIN, T4);
13 (WRITE T3, B, 8);
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.

10.1. Ejercicio 1 (SQL)

Respuesta

Se puede solucionar usando sólo una consulta con una subconsulta, pero es más cómodo y visualmente entendible usando la cláusula WITH. Como se estpa usando agregación en dicha cláusula, el resultado va a ser sólo una fila, y se elige una sola columna: la de marcas, con lo cual resulta un sólo elemento.

```
WITH aumentos_relativos
AS (
    SELECT *, (pr.precio - LAG(pr.precio,1) OVER ( PARTITION BY p.cod_barras_producto
                                                    ORDER BY pr.fecha ASC) aumento
    FROM Productos p INNER JOIN PreciosRelevados pr USING( cod_barras_producto)
    WHERE p.categoría = 'LIMPIEZA'
        AND pr.fecha BETWEEN '01-02-2022' AND '01-02-2023'
    )
SELECT AVG(aumento) inflacion_interanual
FROM aumentos_relativos
```

10.2. Ejercicio 2 (Procesamiento de Consultas)

Respuesta

1. El costo de la primera selección, al ser simplemente un *file scan*, es:

$$C_1 = B(\text{Interpretaciones})$$

$$C_1 = 2,000$$

Resultan de esta instrucción $n_1 = 1,000$ tuplas suponiendo 20 ciudades con distribución uniforme de tuplas; que entran en $B_1 = 100$ bloques.

2. En la junta, usando el índice de clustering provisto y sabiendo que los resultados de la primera operación están en pipeline, el costo de realizar la junta se reduce a

$$C_2 = n(1) * (H(I(\text{personaje}, \text{Interpretaciones}) + \lceil \frac{B(\text{Interpretaciones})}{V(\text{personaje}, \text{Interpretaciones})} \rceil))$$
$$C_2 = 1000 * (3 + \lceil \frac{2000}{100} \rceil))$$
$$C_2 = 23,000$$

Suponiendo que en *Viareggio* están presentes los 100 personajes repartidos entre las 1000 personas dentro de esa ciudad; hay 10 personas en *Viareggio* con el mismo personaje, que podrían coincidir en la junta con las 200 otras personas que usan el mismo personaje en otras ciudades. Entonces, en el peor caso, resultarían 20.000 tuplas de esta etapa.

3. Finalmente, en la proyección no hay costo asociado de acceso a disco, pues se los resultados de las etapas previas se procesan en pipeline.

$$C_3 = 0 \tag{6}$$

Además, como se están seleccionando las personas de la ciudad de *Venecia*; y considerando el peor caso, en el cual todas las personas de *Viareggio* y de *Venecia* tengan la misma distribución de personajes, entonces habría 1000 tuplas como resultado de esta etapa.

Luego el costo total es:

$$C_T = C_1 + C_2 + C_3 = 2,000 + 23,000 = 25,000 \tag{7}$$

10.3. Ejercicio 3 (NoSQL)

Respuesta

a) Respuesta

F Como el particionado de las colecciones se realiza en base a una *shard key*, necesariamente cada shard tiene que tener un atributo identificador relacionado con los documentos que contiene. De esta forma le permite al servidor hacer las consultas de manera directa, sin necesariamente pasar por todos los nodos.

b) Respuesta

F Todas las operaciones de escritura sobre un shard se realizan en el master. Los slaves sólo sirven de respaldo.

c) Respuesta

V Dado que la sharding key no está relacionada al atributo de la consulta, los routers tendrán que consultar en todos los shards que tenga a su cargo, haciendo la consulta menos eficiente.

d) Respuesta

F Si bien la consulta se hace por un valor del atributo de sharding y no es necesario consultar otros shards además de los que contienen dicho valor, no necesariamente el resultado es un único documento.

Podría pasar que el sharding se realice en base a un atributo "localidad". Entonces habría varios documentos que coincidan, haciendo que el resultado no necesariamente sea un único documento.

10.4. Ejercicio 4 (NoSQL)

Respuesta

Respuesta

V Como la estructura C_0 se mergea con el B-tree C_1 , se genera una nueva estructura C_1^* y se borran C_0 (de memoria) y C_1 (de disco).

Respuesta

V Cassandra **no** chequea claves foráneas. Si en C_0 existe un par (k, v) que ya estaba en memoria, lo pisa.

Respuesta

F Los datos en memoria son backupeados en el log, por lo que si ocurre algún desperfecto o falla, se puede recuperar la información.

Respuesta

V Si no existe un dato en memoria (C_0), igualmente podría estar en disco (C_1), por lo que el gestor lo busca en disco si no lo encuentra en memoria.

Respuesta

F Para C_0 suelen usarse AVL's o 2-3 trees, aunque cualquier estructura es posible.

10.5. Ejercicio 5 (Concurrencia y Transacciones)

Respuesta

Respuesta

	T_6	T_8	T_9	X	Y	Z
Valores inic.	$TS = 6$	$TS = 8$	$TS = 9$	read_TS=0 write_TS=0	read_TS=0 write_TS=0	read_TS=0 write_TS=0
	$R(Y)$			-	read_TS(Y)=6	-
		$W(X)$		write_TS(X)=8	-	-
			$W(Z)$	-	-	write_TS(Z)=9
			$R(X)$	read_TS(X)=9	-	-
	$R(Z)$			-	-	read_TS(Z)=8

En el último paso sucede *read too late*: la transacción T_9 escribió el ítem Z y luego una transacción con timestamp menor T_8 quiso leerlo, por lo que habría que abortar T_8 .

10.6. Ejercicio 6 (Recuperación)

Respuesta

Como lo primero que encuentra el algoritmo al recorrer hacia atrás el *log* es un END CKPT entonces sólo deberá retroceder hasta su BEGIN CKPT .

Luego vemos que T_2 commiteó pero T_3 y T_4 no, por lo que hay que deshacer las instrucciones de T_3 y T_4 en disco, escribiendo:

1. C=9 por T_4 línea 14
2. B=8 por T_3 línea 13
3. A=8 por T_3 línea 08

Finalmente se escribe en el log las instrucciones (ABORT, T_3) y (ABORT, T_4) y se vuelca a disco.

11. Coloquio 2023-02-15

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 15 de febrero de 2023

TEMA 20222C3			
CRT	NoSQL	Proc.	
SQL	CyT	DR	
Nota: _____ <input type="checkbox"/> Aprobado <input type="checkbox"/> Insuficiente			
Padrón: _____ Apellido: _____ Nombre: _____ Cantidad de hojas: _____			

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. (CRT)

*"Los amigos de mis amigos
son mis amigos."*

Marco Sacherberg quiere evaluar la exactitud del conocido refrán, y para ello utilizará datos de una red social a la que tiene acceso. Dispone en particular de las siguientes tablas:

- **Usuarios**(cod_usuario, nombre, localidad)
- **Amistades**(cod_usuario_1, cod_usuario_2)

En esta última tabla, cada relación de amistad a ~ b se guarda en ambos sentidos, es decir, como par (a, b) y como par (b, a) . Escriba una consulta en Cálculo Relacional de Tuplas que permite encontrar los códigos de usuario de las personas que cumplen con que “los amigos de sus amigos son también sus amigos” sin excepciones.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

3

TEMPERATURAS_FEB	
n(Temperaturas_Feb) = 400.000	
B(Temperaturas_Feb) = 20.000	
V(localidades, Temperaturas_Feb) = 804	
Histograma de localidades:	
Ciudad de Buenos Aires: 60.000	
Córdoba: 50.000	
Rosario: 35.000	
Bahía Blanca: 33.000	
Otras: 222.000	

Se pide:

- Teniendo la anterior información, y asumiendo que no hay problemas de memoria, estime el costo del plan de ejecución.
 - ¿Cuánta memoria estima que debería tener como mínimo para poder realizar el ordenamiento en memoria sin utilizar disco? Justifique su respuesta.
4. (*SQL*) Al ejecutar la consulta del *Ejercicio 3*, los científicos del *Servicio Meteorológico Nacional* encontraron que un mismo día es devuelto repetidas veces, dado que en la Ciudad de Buenos Aires existen numerosas estaciones meteorológicas, que además toman muchas mediciones por día. Escriba una consulta en lenguaje SQL que devuelva el ranking de los 10 días más calurosos registrados en la Ciudad de Buenos Aires, mostrando para cada uno de esos días la temperatura máxima registrada en alguna estación de la ciudad. Devuelva como resultado únicamente los campos de fecha y de temperatura máxima registrada.
5. (*Concurrencia y Transacciones*) Suponga que en una base de datos con un cierto estado inicial y sin transacciones en curso comienzan a ejecutarse en forma solapada tres transacciones T_1, T_2 y T_3 . Indique si las siguientes afirmaciones relativas a dicha ejecución concurrente son verdaderas ó falsas, justificando su respuesta.
- Si el solapamiento es equivalente por conflictos a la ejecución serial $T_1 \rightarrow T_3 \rightarrow T_2$, entonces no puede ser equivalente por conflictos también a la ejecución serial $T_2 \rightarrow T_3 \rightarrow T_1$.
 - Si el solapamiento es equivalente por resultado a la ejecución serial $T_1 \rightarrow T_3 \rightarrow T_2$, entonces es también equivalente por conflictos a dicha ejecución serial.
 - Si el grafo de precedencias resultante de la ejecución incluye el arco dirigido $T_2 \rightarrow T_3$, entonces no puede incluir el arco dirigido $T_3 \rightarrow T_2$.
 - Si cada vez que una transacción utiliza un recurso lo hace manteniendo un *lock exclusivo* sobre ese recurso, entonces el solapamiento será serializable.
 - Si el grafo de precedencias resultante de la ejecución es *acíclico*, el mismo puede tener más de un ordenamiento topológico.
- Nota: Suponga que en principio desconocemos si el SGBD garantiza la serializabilidad ó no.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

2. (*NoSQL*) Luego de repasar la lógica del cálculo de tuplas, Marco Sacherberg decide cargar los datos de la red social a la que tiene acceso dentro de una base Neo4j. En esta base, Marco colocó un nodo por cada usuario, y un arco por cada amistad entre usuarios, siguiendo la siguiente nomenclatura:

```

1 CREATE (u1: Usuario {cod_usuario: '01981571', nombre: 'Alberto Vulnek', localidad: 'Boulogne'});
2 CREATE (u2: Usuario {cod_usuario: '03134990', nombre: 'Claudia Remmini', localidad: 'Pilar'});
3 MATCH (u1: Usuario {cod_usuario: '01981571'}), (u2: Usuario {cod_usuario: '01981571'});
4 CREATE (u1)-[:AMIGO_DE]-(u2);
5
6
    
```

Escriba una consulta en *Cypher* que devuelva el listado de los usuarios para los cuales se cumple que “los amigos de sus amigos son también sus amigos”.

Nota: Le sugerimos utilizar la estructura `MATCH ... WHERE [NOT] EXISTS { MATCH pattern } ...`

3. (*Procesamiento de consultas*) El último domingo se registró en la Ciudad de Buenos Aires el día de febrero más caluroso de los últimos 61 años. Para corroborar esta información, el *Servicio Meteorológico Nacional* utilizará la siguiente tabla en la que almacena todos los registros de temperatura históricos del mes de febrero:

- **Temperaturas_Feb**(cod_estación, fecha, hora, latitud, longitud, localidad, temperatura)

Para encontrar los registros más altos de temperatura ocurridos en Buenos Aires durante el mes de febrero, se ejecutará la siguiente consulta:



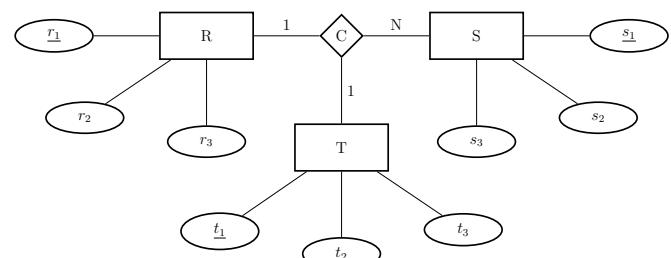
El operador τ del plan de ejecución representa el ordenamiento de la tabla por el atributo ‘temperatura’. Dicho ordenamiento se realiza en forma similar al ordenamiento de una tabla en una proyección.

La tabla dispone de un índice de clustering de tipo árbol por *localidad*, de altura 5. Además, puede considerar para sus cálculos la siguiente información de catálogo:

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

4

6. (*Diseño Relacional*) Considere el siguiente modelo entidad-interrelación:



Este modelo se traduce a una base de datos relacional compuesto por las siguientes 4 relaciones:

- **R**(r₁, r₂, r₃)
- **S**(s₁, s₂, s₃)
- **T**(t₁, t₂, t₃)
- **C**(r₁, s₁, t₁)

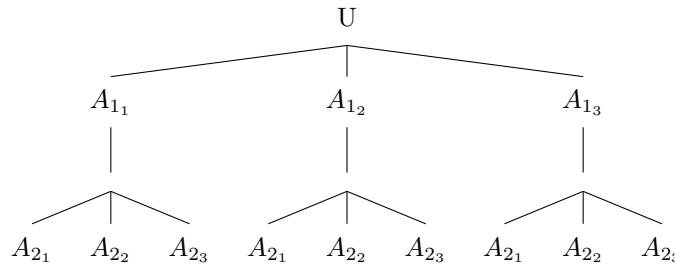
Si se quisiera respetar las restricciones impuestas por el modelo conceptual, entonces (indique V o F, y justifique su respuesta):

- La clave primaria de la relación C debe ser el atributo *s₁*.
- Debe regir la dependencia funcional $(s_1, t_1) \rightarrow r_1$.
- Debe regir la dependencia funcional $(s_1, r_1) \rightarrow t_2$.
- El atributo *s₁* en la relación S debe ser configurado como clave foránea que hace referencia a C, para asegurar que toda instancia de S se vincule con al menos una instancia de R.
- No debe haber nunca dos tuplas de C con igual valor de *t₁*.

11.1. Ejercicio 1 (CRT)

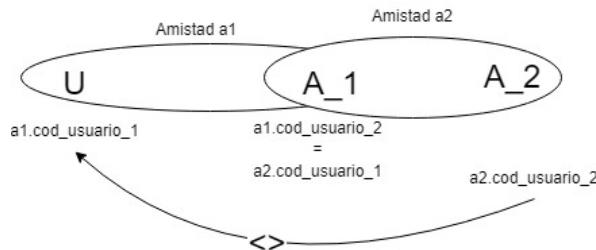
Respuesta

Pensando en un usuario U que tiene amistades A_1 , para verificar el ejercicio es necesario recurrir a los amigos de los amigos A_2 del usuario.



Se puede pensar "quiero a los usuarios que tengan amistades ($\exists a_1$) cuyos amigos ($\forall a_2$) también sean mis amigos"

Usuario= U Amigo1=A_1 Amigo2=A_2



$\{u.cod_usuario |$ $Usuarios(u) \wedge (\exists a_1)(Amistades(a_1) \wedge$
 $a_1.cod_usuario_1 = u.cod_usuario \wedge$
 $(\forall a_2 \in Amistades)(a_2.cod_usuario_1 = a_1.cod_usuario_2 \wedge$
 $a_2.cod_usuario_2 \neq u.cod_usuario))\}$

11.2. Ejercicio 2 (NoSQL)

Respuesta

```

MATCH (u:Usuario-[:AMIGO_DE]->u2:Usuario-[:AMIGO_DE]->u3:Usuario)
WHERE EXISTS{
    MATCH (u-[:AMIGO_DE]->u3-[:AMIGO_DE]->u2),
    WHERE u3 <> u}
RETURN u
  
```

11.3. Ejercicio 3 (Procesamiento de Consultas)

a) Respuesta

Usando el índice de clustering, la selección se podría hacer con un index scan, cuyo costo sería

$$C_1 = H(localidad, Temperaturas_Feb) + \lceil \frac{B(Temperaturas_Feb)}{V(localidades, Temperaturas_Feb)} \rceil \quad (8)$$

$$C_1 = 5 + \lceil \frac{20,000}{804} \rceil \quad (9)$$

$$C_1 = 30 \quad (10)$$

Luego, suponiendo que hay suficiente memoria, el resultado no se volvería a cargar en disco sino que se procesaría por la siguiente etapa una vez finalizado. Por lo tanto, la etapa de ordenamiento no tendría ningún costo.

$$C_2 = 0 \quad (11)$$

Y el costo total del plan de ejecución sería C_1 .

b) **Respuesta**

La mínima cantidad de memoria estima que debería tener como mínimo para poder realizar el ordenamiento en memoria sin utilizar disco sería $M_{min} = n(\sigma_{localidad='CiudaddeBuenosAires'})/F(Temperaturas_Feb) = 3000$, para que entren todos los resultados de la selección y se pueda realizar el ordenamiento.

11.4. Ejercicio 4 (SQL)

Respuesta

Se puede solucionar agrupando por fecha (particiones) y tomando el máximo de cada una. Como luego hay que tomar el ranking de los 10 días más calurosos y ya se usó la cláusula de múltiples particiones, se puede usar paginación (OFFSET ... ROWS FETCH FIRST ... ROWS ONLY)

```
SELECT DISTINCT t.fecha, t.temperatura
FROM temperaturas_feb t
WHERE t.localidad= 'Ciudad de Buenos Aires'
AND t.temperatura=
    SELECT MAX(t1.temperatura)
    FROM temperaturas_feb t1
    WHERE t1.localidad = t.localidad
        AND t1.fecha = t.fecha
)
ORDER BY t.temperatura DESC
OFFSET 0 ROWS FETCH FIRST 10 ROWS ONLY;
```

11.5. Ejercicio 5 (Concurrencia y Transacciones)

a) **Respuesta**

V Como el solapamiento es equivalente a la ejecución serial $T_1 \rightarrow T_3 \rightarrow T_2$, sus conflictos vienen dados por ese orden: $T_1 \rightarrow T_3$, $T_3 \rightarrow T_2$. En la ejecución serial $T_2 \rightarrow T_3 \rightarrow T_1$ no se respetan de a pares los ordenes (no aparecen las transacciones en el mismo orden que la ejecución serial previa), por lo que no serían equivalentes.

b) **Respuesta**

F La implicancia entre equivalencias es conflictos \Rightarrow resultados, no a la inversa. La primera no depende del estado inicial de la base de datos. Es la más fuerte de las tres. Mientras que la segunda sí depende del estado inicial de la base.

c) **Respuesta**

V Suponiendo que se quiere un resultado serializable, incluir el arco $T_3 \rightarrow T_2$ implicaría formar un grafo cíclico, haciendo que **no** sea serializable.

d) **Respuesta**

F Sin ningún otro método o procedimiento, como el uso de S2PL o R2PL, no se puede garantizar la serializabilidad, porque el lock exclusivo podría adquirirse justo antes de realizar la instrucción I_i e inmediatamente después *unlockearlo*.

e) **Respuesta**

V Como el orden topológico se forma eliminando los nodos que no poseen predecesores en forma recursiva y de a uno a la vez, hasta que no quede ninguno, el orden en que los nodos fueron eliminados podría ser distinto de acuerdo al primer nodo sin predecesores que se elija.

subsectionEjercicio 6 (Diseño Relacional)

a) **Respuesta**

V Dado que la relación es $(1, 1, N)$, basta con indicar el valor de s_1 para reconocer los de t_1 y r_1 .

b) **Respuesta**

No necesariamente. Puede reemplazarse por la dependencia funcional $s_1 \rightarrow r_1$ o mejor dicho $s_1 \rightarrow r_1, t_1$, sin que r_1 sea necesariamente determinante de la dependencia funcional.

c) **Respuesta**

No necesariamente. Por los puntos previos, la dependencia es con t_1 por ser clave de T , pero no hay restricciones sobre t_2 . Podría haber valores distintos (s_{11}, r_{11}, t_{21}) y (s_{11}, r_{11}, t_{22}) y de igual manera respetar el diagrama original.

d) **Respuesta**

No es imposible, pero en general se indica a s_1 como clave foránea en C .

e) **Respuesta**

Un par de atributos (t_1, r_1) pueden estar relacionados con varios valores de s_1 en C . Como lo indica la relación del punto b), para identificar a un t_1 sólo hace falta (s_1, r_1) . Al no contradecir ninguna restricción, puede haber dos valores repetidos de t_1 .

12. Coloquio 2022-12-21

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

1

Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 21 de diciembre de 2022

TEMA 20222C2				
CRT	Proc.	SQL		
NoSQL	CyT	Rec.		
Nota: <input type="checkbox"/> Aprobado <input type="checkbox"/> Insuficiente				
Padrón: _____ Apellido: _____ Nombre: _____ Cantidad de hojas: _____				

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. (*Cálculo Relacional de Tuplas*) Las siguientes tablas indican la cantidad de ejemplares de distintas especies presentes en cada una de las reservas naturales del país:

- Reservas(nombre_reserva, superficie, año_creación)
- Ejemplares(nombre_reserva, nombre_especie, cantidad.ejemplares)

Escriba una consulta en Cálculo Relacional de Tuplas que encuentre el nombre y la cantidad de ejemplares de la especie “tatú carreta” presentes en aquellas reservas que tienen ejemplares de “tatú carreta” pero que no tienen ningún ejemplar de “zorro gris”.

Nota: Asuma que el atributo cantidad.ejemplares sólo puede tomar valores mayores a cero.

Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires) 3

4. (*NoSQL*) La nueva red social *Rhinos* publicó una nueva API que permite consultar su base de datos en MongoDB, en la que cada posteo tiene la siguiente estructura en formato JSON:

```

1 {  
2   post_id: "1928500576325.104344819840",  
3   user_id: 2401740,  
4   user_name: "Lionel Messi",  
5   picture: "3f2103ab1a20299ee0.jpg",  
6   texto: "Buen día!",  
7   tags: ["futbol", "copa", "qatar2022"],  
8   likes: ["jorge", "celia"]  
9 }
  
```

Lionel quiere revisar a cuántos de sus posteos etiquetados como “futbol” les dieron likes su padre “jorge” y su madre “celia”. Quiere la respuesta en el siguiente formato:

```

1 {  
2   nombre : "jorge",  
3   cantidad : 18 },  
4 {  
5   nombre : "celia",  
6   cantidad : 20 }
  
```

Escriba la consulta en MongoDB que permita a Lionel obtener dicho resultado.

5. (*Concurrencia y transacciones*) Para cada una de las siguientes afirmaciones, indique si la misma es verdadera ó falsa, justificando su respuesta:

- Si se utiliza el protocolo de lock de dos fases (2PL) para el control de concurrencia, no puede ocurrir la anomalía del fantasma.
- Si se utiliza el protocolo de lock de dos fases (2PL) para el control de concurrencia, es posible que ocurra la anomalía de la lectura sucia.
- Bajo el mecanismo de control de concurrencia *Snapshot Isolation* no es posible que ocurra la anomalía del fantasma.
- Bajo el mecanismo de control de concurrencia *Snapshot Isolation* nunca resulta necesario abortar una transacción para asegurar la serialización.

6. (*Recuperación*) Un SGBD implementa el algoritmo de recuperación UNDO/REDO con checkpoint activo. Indique si las siguientes afirmaciones sobre el funcionamiento del algoritmo son verdaderas ó falsas, justificando su respuesta:

- Los ítems de datos modificados por una transacción T_i deben ser *flushed* a disco antes de escribir (*COMMIT*, T_i) en el archivo de *log*.
- Cuando se modifica un ítem de datos X es necesario registrar en el *log* tanto su valor anterior como su nuevo valor.
- Cuando se modifica un ítem de datos X es obligatorio *flush* el nuevo valor de X a disco antes de *flush* el registro de *log* correspondiente a disco.
- Si el sistema se reinicia y el algoritmo detecta que una transacción T_i no había llegado a *commitear*, se debe escribir (*ABORT*, T_i) en el archivo de *log* y *flusharlo* a disco antes de deshacer las modificaciones realizadas por T_i en disco.

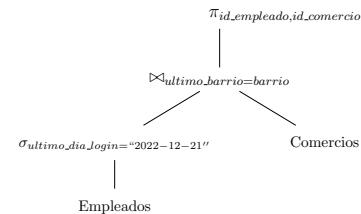
Departamento de Computación (Facultad de Ingeniería - Universidad de Buenos Aires)

2

2. (*Procesamiento de consultas*) La empresa de envío de pedidos *Lentti* analiza constantemente en qué barrios se encuentran sus empleados, para decidir a qué comercios enviarlos. En la tabla *Empleados* registra, entre otros datos de sus empleados, el último barrio en el que estuvieron y el último día en que activaron la aplicación. También mantiene una tabla con la información de los comercios que venden productos a través de su aplicación:

- *Empleados(id_empleado, DNI, nombre, último_barrio, último_dia_login, calif)*
- *Comercios(id_comercio, nombre, dirección, barrio, calificación, tipo_contrato)*

El gerente de despachos quiere ver en la cercanía de qué comercios se encuentra cada empleado activo. Para ello, considera activo a un empleado que se logró en el día y considera que un empleado está cerca de un comercio si está en el mismo barrio. De esta forma, ejecuta la siguiente consulta sobre la base de datos:



La base de datos dispone de un índice de tipo árbol por *id.empleado* en *Empleados* y de un índice por *id.comercio* en *Comercios*. Además, puede considerar para sus cálculos la siguiente información de catálogo:

EMPLEADOS	COMERCIOS
n(Empleados) = 150.000	n(Comercios) = 30.000
B(Empleados) = 30.000	B(Comercios) = 3.000
H(I(id_empleado, Empleados)) = 3	H(I(id_comercio, Comercios)) = 3

También se sabe que cada día están activos alrededor de un 10% de los empleados.

Se pide:

- Teniendo la anterior información, y asumiendo que no hay problemas de memoria, estime el costo de la consulta eligiendo los métodos óptimos.
- Indique qué ocurriría si hubiera únicamente 502 bloques de memoria disponibles.

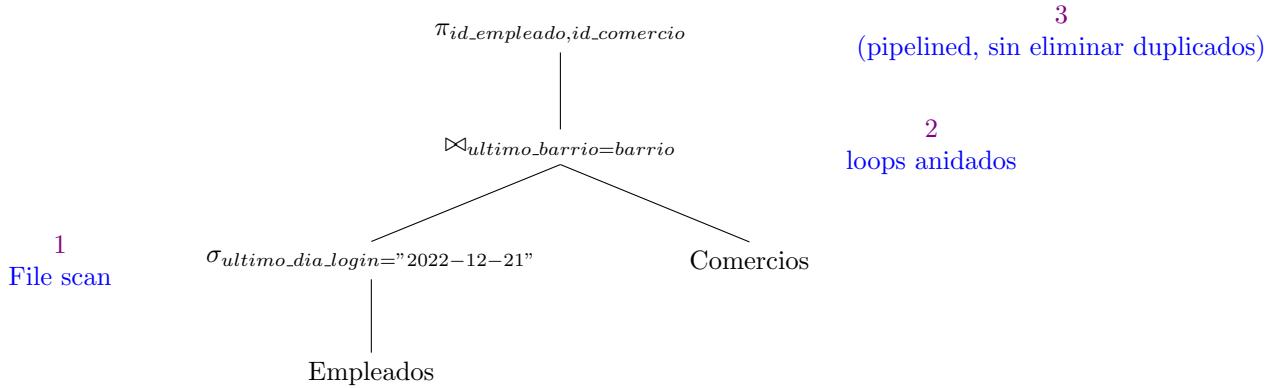
3. (*SQL*) Para las mismas tablas del *Ejercicio 2*, escriba una consulta en SQL que encuentre los nombres de los barrios en los cuales no hay ningún comercio con calificación mayor o igual a 8.

12.1. Ejercicio 1 (CRT)

$$\{e.nombre_reserva, e.cantidad_ejemplares \mid \begin{aligned} & Ejemplares(e) \wedge \\ & e.nombre_especie = 'tatu carreta' \wedge \\ & (\nexists e_2)(Ejemplares(e_2) \wedge \\ & e_2.nombre_reserva = e.nombre_reserva \wedge \\ & e_2.nombre_especie = 'zorro gris') \} \}$$

12.2. Ejercicio 2 (Procesamiento de Consultas)

Respuesta



Suponiendo que:

- No hay problemas de memoria
- los índices brindados son primarios
- cada día están activos alrededor de un 10% de los empleados $\Rightarrow \sigma_{ultimo_dia_login='2022-12-21'}$ tiene $n = 15,000$ tuplas que cumplen

1. Al no haber un índice por el cual buscar por fecha, el costo de la instrucción 1 es simplemente $B(Empleados)$

$$C_1 = B(Empleados)$$

$$C_1 = 30,000$$

Resultan de esta instrucción $n_1 = 15,000$ tuplas por el tercer supuesto, que entran en $B_1 = 3000$ bloques.

2. En el paso 2 se puede utilizar o la junta por loops anidados o el método de *sort-merge*, pues ambos tienen el mismo costo al disponer de memoria ilimitada, aunque prefiriendo el segundo por su menor costo computacional. En ambos casos, el costo de realizar la junta se reduce a

$$C_2 = B(Comercios)$$

$$C_2 = 3,000$$

que sería el costo de cargar la tabla en memoria. Luego se haría el quicksort de cada tabla por el atributo de junta y por último se realizaría la junta en memoria.

3. Finalmente, en el paso 3 no hay costo asociado de acceso a disco, pues se los resultados de las etapas previas ya se encuentran en memoria.

$$C_3 = 0 \tag{12}$$

Luego el costo total es:

$$C_T = C_1 + C_2 + C_3 = 30,000 + 3,000 = 33,000 \tag{13}$$

En caso de que hubiera 502 bloques, se podría realizar una junta **Hash GRACE** y *pipeline* la etapa de selección y junta. De esta manera, los resultados de la selección pasan directamente a hashearse y tampoco habría costo por la proyección.

12.3. Ejercicio 3 (SQL)

Respuesta

```
SELECT c.barrio
FROM Comercios c
WHERE NOT EXISTS ( SELECT *
                    FROM Comercios c2
                    WHERE c.calificacion >7)
```

12.4. Ejercicio 4 (NoSQL)

Respuesta

```
[{"$unwind": "$grades"}, {"$match": {"user_id": "2401740", "tags": "futbol", "likes": {"$in": ["jorge", "celia"] } } }, {"$group": {"_id": "$likes", "nombre": {"$first": "$like"}, "cantidad": {"$sum:1"}}, {"$project": { "_id" : 0, "nombre":1,"cantidad":1}}]
)
```

12.5. Ejercicio 5 (Concurrencia y Transacciones)

a) **Respuesta**

F Bajo R2PL, donde los locks sólo pueden ser liberados después del commit si es verdadero, pero en 2PL donde los recursos se liberan una vez la transacción no lo vaya usar más, no se garantiza que no ocurra la anomalía del fantasma.

b) **Respuesta**

F Si bien se necesita tener el lock sobre un recurso cuando es escrito, bajo el protocolo 2PL sólo se puede liberar una vez cierta transacción T_i no vaya a usar más ese recurso, pero puede suceder que lo termine de usar, no haya committeado, otra transacción T_j lea ese recurso, e inmediatamente después se aborte T_i .

c) **Respuesta**

V No puede ocurrir, porque la transacción ve la misma snapshot a lo largo de su vida.

d) **Respuesta**

F Si bien cuando dos transacciones intentan modificar un mismo ítem de datos, generalmente gana aquella que hace primero su commit y la otra se aborta; para garantizar serializabilidad no basta sólo el método de control de concurrencia de *Snapshot Isolation*, sino que se debe combinar con la validación permanente con el grafo de precedencias buscando ciclos de conflictos RW, y con locks de predicados en el proceso de detección de conflictos, para detectar precedencias

12.6. Ejercicio 6 (Recuperación)

Respuesta

a) **Respuesta**

F Los ítems modificados pueden ser guardados en disco antes o después de hacer el *commit*, pues se cumplen ambas reglas: WAL y FLC.

b) **Respuesta**

V Como este algoritmo funciona guardando las instrucciones de cada transacción en el log de manera inmediata en disco, pero modifica los valores de los ítems correspondientes en cualquier momento; es necesario tener los valores viejos y nuevos de cada ítem modificado para poder rehacer las instrucciones que commitearon y deshacer las que no lo hicieron.

c) **Respuesta**

F En ningún algoritmo se *flushea* el valor del ítem modificado a disco antes que su correspondiente instrucción en el log, porque si pasa algo justo antes de cargar en disco la instrucción del log pero después de la modificación del ítem en disco, no queda registro de ese cambio y se rompe el algoritmo.

d) **Respuesta**

F El orden correcto sería el inverso. Dado que estos algoritmos son idempotentes, si se vuelve a reiniciar el sistema cuando se estaba realizando la recuperación de otro reinicio, el sistema va a quedar en un estado estable de igual manera.

Suponiendo que primero se escribe el ABORT de la instrucción T_i en el log y **luego** se hace su UNDO, podría pasar que justo antes de hacer el UNDO se reinicia el sistema y queda registrado el ABORT en el log, pero es inconsistente con los valores de los ítems modificados en disco por esa transacción T_i , que no pudieron deshacerse.

13. Apéndice

13.1. (CRT) Caso de uso de \forall . Ejemplo de teórica

En un ejemplo de la teórica se analiza el uso del \forall en el caso de encontrar al jugador el mundial de mayor edad.

$$\{p.name | \ Player(p) \wedge (\forall p_2)(\sim Player(p_2) \vee p.birth_date \leq p_2.birth_date)\}$$

Al usar el \forall , ésta expresión es correcta porque sólo se queda con los p_2 que sean jugadores y cumplan con la condición, y si por cualquier razón p_2 resulta ser un elemento que no pertenece al dominio (en este caso jugadores), no arruina a p que sí es jugador.

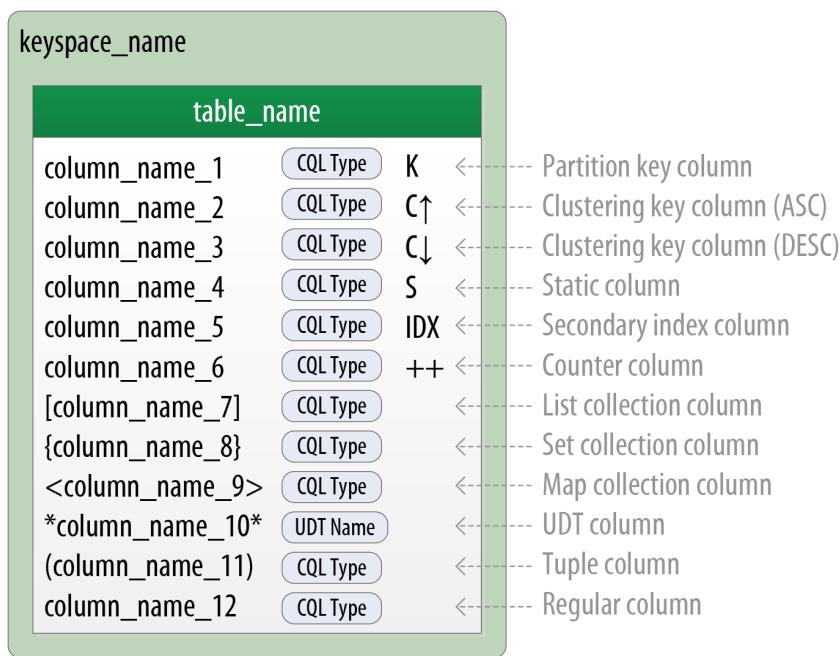
Suponiendo un caso en el que se toman p como Jugador y p_2 como Contador. Al no ser p_2 un Jugador, no debería afectar el resultado de p , que sería el jugador de mayor edad, dado que se está comparando un Jugador con Contadores. ✓

En cambio, si se usa la expresión

$$\{p.name | \ Player(p) \wedge (\forall p_2)(Player(p_2) \wedge p.birth_date \leq p_2.birth_date)\}$$

se si se comparan un Jugador con un Contador, la lógica válida sería decir que el jugador de mayor edad es p , sin embargo, al no ser p_2 un Jugador, la expresión total devuelve falso, erróneamente excluyendo a p como el jugador de mayor edad. ✗

13.2. (NoSQL) Ayuda Cassandra - Diagrama Chebotko



13.3. (Concurrencia) Implicancias útiles

- serializabilidad \Rightarrow Aislamiento de transacciones.

Se verifica con el grafo de precedencias

- Recuperable \Rightarrow Serializable ✗

- Serializable \Rightarrow Recuperable ✗

- Evita rollbacks en cascada \Rightarrow Recuperable ✓

- Evita rollbacks en cascada \Rightarrow Serializable ✗

- S2PL, R2PL \Rightarrow Recuperable, Serializable, Evita rollbacks en cascada ✓

13.4. Costos

- HASH GRACE

Las tablas en una junta de este tipo se hashean en base al atributo de junta .

A la función de hash le paso una tupla de R y me devuelve un valor entre $[0, k - 1]$. Así concentra los valores que coinciden $h(R.A_1)$ con $h(S.A_1)$, para dos tablas R, S .

- HASH GRACE

Las filas con el mismo valor de A_1 están en la misma partición R_i o S_i , pero no es cierto que dos filas que estén en la misma partición tengan el mismo valor del atributo.

Calculo Relacional de Tuplas

- **Predicados:** funciones cuyo resultado es un valor de verdad
- **Operaciones** entre predicados (\wedge , \vee , \neg , \rightarrow)
- **Cuantificadores** de variables (\forall , \exists)

Ejemplos

Nombre de países que jugaron el mundial

$$\{ n.name \mid \text{NationalTeams}(n) \}$$

Jugadores nacidos antes de 1980

$$\{ p.name \mid \text{Players}(p) \wedge p.birth_year < 1980 \}$$

Jugadores de la selección española

$$\begin{aligned} \{ p.name \mid \text{Players}(p) \wedge (\exists n)(\\ \text{NationalTeams}(n) \wedge n.short_name = p.national_team \wedge n.name = "Spain") \} \end{aligned}$$

Jugador más anciano

$$\begin{aligned} \{ p.name \mid \text{Players}(p) \wedge (\exists p2)(\\ \neg \text{Players}(p2) \vee p2.birth_year \geq p.birth_year) \} \end{aligned}$$

SQL

Integridad

[ON DELETE SET NULL | RESTRICT | CASCADE | SET DEFAULT]

[ON UPDATE SET NULL | RESTRICT | CASCADE | SET DEFAULT]

Si una fila t se elimina/se modifica su pk, y es referenciada por una fk de otra tabla

- CASCADE: Se del/mod las filas que referencian a t ; luego se del/mod t
- SET NULL: Se ponen en NULL las fk que lo referencian; luego se del/mod t
- RESTRICT: No se modifica/elimina t

With Recursivo

```
WITH RECURSIVE  $T[(A_1, A_2, \dots, A_n)]$ 
AS (<initial_value_query>
      UNION <subquery>
      <query with  $T$ >;
```

WITH recursivo 1: Pseudocódigo

```
 $T_0 = initial\_value\_query(R_1, R_2, \dots);$ 
 $T_{new} = T_0;$ 
do
|    $T = T_{new};$ 
|    $T_{new} = T_0 \cup subquery(T, R_1, R_2, \dots);$ 
while  $T_{new} \neq T;$ 
return query( $T, \dots$ );
```

Ejemplo

Dada la relación Vuelos(codVuelo, ciudadDesde, ciudadHasta) que indica todos los vuelos que ofrece una aerolínea, encuentre todas las ciudades que son 'alcanzables' desde París, independientemente de la cantidad de escalas que sea necesario hacer.

```
WITH RECURSIVE DestinosAlcanzables(ciudad)
AS (VALUES ('Paris')
      UNION
      SELECT v.ciudadHasta AS ciudad
      FROM DestinosAlcanzables d, Vuelos v
      WHERE d.ciudad = v.ciudadDesde
    )
SELECT ciudad FROM DestinosAlcanzables;
```

Funciones de Ventana

Particion Unica

Funciones de Ventana

- RANK(): Devuelve el ranking según el ordenamiento
- ROW_NUMBER(): Como rank pero sin empates (no deterministic)
- LAG(A, offset): Diferencia que toma el att A en una fila a distancia offset hacia atras

```
SELECT RANK() OVER (ORDER BY marca_seg) AS posición,
       nombre_atleta, país_origen, marca_seg,
       marca_seg - LAG(marca_seg, 1) OVER (ORDER BY marca_seg) AS diferencia
FROM Final_2009;
```

Podemos utilizar la cláusula WINDOW si repetiremos la ventana

```
SELECT RANK() OVER ventana_marca_seg AS posición,
       nombre_atleta, país_origen, marca_seg,
       marca_seg - LAG(marca_seg, 1) OVER ventana_marca_seg AS diferencia
FROM Final_2009
WINDOW ventana_marca_seg AS (ORDER BY marca_seg)
ORDER BY diferencia DESC;
```

Múltiples Particiones

Dentro de una ventana, podemos utilizar la cláusula PARTITION BY para seccionar las ventanas y la aplicación de funciones de ventana.

Podemos también aplicar funciones de conjunto sobre las ventanas.

```
SELECT .., [ f(Ai) | w([Ai], ..) ] OVER (PARTITION BY Ak
{ ORDER BY Aj [ ASC | DESC ] }), ..
FROM ...
```

```
SELECT producto,
       RANK() OVER (PARTITION BY producto ORDER BY cantidad DESC) AS posición,
       país, cantidad
  FROM Exportaciones
 ORDER BY producto, posición;
```

```
SELECT cliente, fecha, concepto, monto,
       SUM(monto) OVER (PARTITION BY cliente ORDER BY fecha) AS saldo
  FROM Operaciones
 ORDER BY cliente, fecha;
```

cliente	fecha	concepto	monto	saldo
003	2020-04-01	Saldo inicial	\$10.000	\$10.000
003	2020-04-02	Depósito en efectivo	\$2.000	\$12.000
003	2020-04-18	Adidas	-\$3.500	\$8.500
003	2020-04-23	Transferencia a Jorge Mussi	-\$2.000	\$6.500
004	2020-04-01	Saldo inicial	\$5.000	\$5.000
004	2020-04-15	Farmacia	-\$2.100	\$2.900
004	2020-04-28	Transferencia de Emilce Vega	\$2.800	\$5.700

Observemos que la función **SUM()** utiliza en cada fila un **marco (frame)** que va desde el inicio del mes de ese cliente hasta la fecha actual, calculando así las “sumas parciales”.

No SQL

Objetivos

- Escalabilidad
- Performance
- Flexibilidad
- Capacidad de Distribución

Clasificación

- Clave-Valor
- Orientadas a Documentos
- Wide Column
- Basadas en Grafos

BDD Distribuidas

Fragmentación: dividir un conjunto de agregados entre un conjunto de nodos

- Almacenar conjuntos muy grandes (no entran en 1 solo nodo)
- Paralelizar el procesamiento

Replicación: almacenar múltiples copias de un mismo dato en distintos nodos

- Mecanismo de backup
- Repartir la carga de procesamiento
- Genera un problema de consistencia

Modelos de Consistencia

- **Secuencial:** resultado equivalente al de alguna ejecución secuencial
- **Causal:** capturar eventos que puedan estar causalmente relacionados
- **Eventual**

CAP Theorem

Es imposible garantizar simultáneamente más de dos de los siguientes:

- Consistencia - Consistency
- Disponibilidad - Availability
- Tolerancia a Particiones - Partition Tolerance

Documentos: MongoDB

Selección

```
collection.find( query, projection ).sort( order ).limit( n )
```

- **Operadores de Consulta:** and, or, eq, neq, gt, gte, lt, lte, in, nin, elemMatch, size
- **Sort:** { 'field': 1 | -1 } (ASC | DESC)

Agregación

```
collection.aggregate( [ {stage_1}, {stage_2}, ..., {stage_N} ] )
```

- **Match:** { \$match: { query } }
- **Project / AddFields:** { \$project: { field: 0 | 1 | <expr> } }
- **Group:** { \$group: { _id: <expr>, [fields]: <accumulator> } }
Acumuladores: sum, min, max, first, last, add, addToSet, count
- **Sort:** { \$sort: { field: 1 | -1 } }
- **Limit / Skip / Sample:** { \$limit: n }
- **Unwind:** { \$unwind: field } -> deconstruye según un campo de lista

ABM

- collection.insertOne(doc)
- collection.insertMany([doc, doc, ...])
- collection.deleteOne(filter)
- collection.deleteMany(filter)
- collection.updateOne(filter, update, {upsert: <bool>})
- collection.updateMany(filter, update, {upsert: <bool>})

Filter: { field: <expr> }

Update: {

```
$set: { field: <expr> }, $unset: { field: 1 },
$inc: { field: n }, $min: { field: val, ... }, $max: { field: val, ... }
}
```

Grafos: Neo4J

MATCH (nombre:Tipo) [WITH] **WHERE** ... **RETURN** ... **ORDER BY** ... **LIMIT**

Podemos indicar la dirección de una relación (->) o no (-)

Con un * en una interrelación indicamos indeterminados saltos

Funciones de string: STARTS WITH, ENDS WITH, CONTAINS

Comparación de regex: ~= (eg. a.name!= "A.*")

Agregacion: min, max, sum, count, avg, collect

Ejemplos

Enemigos con amigos en común

```
MATCH (n:Persona)-[:AMIGO]-(m:Persona),  
      (m:Persona)-[:AMIGO]-(o:Persona),  
      (n:Persona)-[:ENEMIGO]-(o:Persona)  
RETURN DISTINCT n.nombre, o.nombre
```

Amigos de distancia entre Juan y Luis

```
MATCH ( juan:Persona {nombre: 'Juan'} ), ( luis:Persona {nombre: 'Luis'} ),  
      p=(juan:Persona)-[:AMIGO*]-(luis:Persona)  
RETURN length(p)  
ORDER BY length(p)  
LIMIT 1
```

Primos de Victor

```
MATCH (victor:Persona {nombre:'Victor'}),  
      (primo:Persona)-[:HIJO]->(tio:Persona)-[:HIJO]->  
      (abuelo:Persona)<-[HIJO]-(padre:Persona)<-[HIJO]-(victor)  
WHERE padre<>tio  
RETURN DISTINCT primo.nombre
```

Ancestros de Victor

```
MATCH (victor:Persona {nombre:'Victor'}),  
      (ancestro:Persona)<-[HIJO*]-(victor)  
RETURN DISTINCT p.nombre
```

Transacciones

ACID

- Atomicidad - Atomicity
- Consistencia - Consistency
- Aislamiento - Isolation
- Durabilidad - Durability

Anomalias

Generan conflictos ReadWrite(RW), WriteRead(WR), etc

- Dirty Read (WR): Se lee una modificación que luego se aborta
- Lost Update (RW): Se modifica un elemento leído por una transacción en curso
- Dirty Write (WW): Se modifica un elemento ya modificado y se aborta
- Phantom: Se observa un conjunto que cumple una condición, que luego cambia

Nivel de aislamiento	Lectura sucia	Lectura no repetible	Fantasma
READ UNCOMMITTED	✗	✗	✗
READ COMMITTED	✓	✗	✗
REPEATABLE READ	✓	✓	✗
SERIALIZABLE	✓	✓	✓

Serializabilidad

Un solapamiento de Tx es **serializable** si es equivalente a una ejecución **serial**

Grafo de precedencias

Equivalente por **conflictos**

- Se crea un nodo por cada Tx
- Se agrega un arco por cada conflicto RW, WR o WW
- No debe haber ciclos

Control de Conurrencia

Enfoques: Pesimista (garantizar que no haya conflictos) / Optimista (deshacerlos)

Basada en Locks

Locks [L / U] o Locks de Lectura-Escritura [LR / LW / U]

2PL (Lock de 2 fases): No se puede adquirir un lock luego de liberar alguno

- Garantiza serializabilidad
 - Puede generar deadlocks o livelocks (inanición)

2PLC (Conservador): Cada Tx adquiere todos los locks necesarios al principio

- Previene deadlocks

Detección de deadlocks: Grafo de alocación y Lock Timeout

Prevención de livelock: prioridad a Tx más antiguas (que no aborte siempre la misma)

Crabbing Protocol: Al recorrer el Árbol B+, liberar los nodos padres si es seguro

Basada en Timestamps

Se asigna un timestamp a cada Tx y se permiten conflictos ordenados.

Se mantiene para cada ítem X:

- **read_TS(X)**: mayor timestamp de lectura (mayor TS == más joven)
 - **write_TS(X)**: mayor timestamp de escritura

Se aborta la Tx más antigua en casos de “read/write too late”

Thomas's Write Rule: si en un caso de “write too late”, ninguna Tx posterior leyo el valor, se puede descartar, sin abortar, de manera segura. (serializabilidad por vistas)

Snapshot Isolation

Cada Tx ve un ‘snapshot’ de la db al instante de su inicio.

Permite mayor solapamiento. Requiere mayor espacio y deshacer conflictos WW.

First-Committer-Wins: Ante conflictos, gana la Tx en commitear primero.

Para garantizar la serializabilidad es necesario validar permanentemente el grafo de dependencias buscando conflictos RW + Locks de predicados en el proceso, para detectar precedencias.

Recuperacion

Un solapamiento es **recuperable** si una Tx no commitea hasta que todas las Tx que hayan modificado datos que haya leído hayan commiteado. (!= serializable).

=> No será necesario revertir Tx commiteadas

Se almacena en un log:

- BEGIN, T
- WRITE, T, X, x_old, [x_new]
- COMMIT, T
- ABORT, T

Puede haber **rollbacks en cascada** si se leen valores no commiteados

- S2PL (Strict 2PL): los locks de escritura sólo pueden ser liberados después de haber commiteado la transacción
- timestamps: se puede bloquear a los lectores hasta que commiten los escritores

Reglas

Write Ahead Log (WAL): antes de guardar un ítem modificado en disco, se debe escribir el registro de log correspondiente, en disco

Force Log at Commit (FLC): antes de realizar el commit el log debe ser volcado a disco.

Algoritmos de Recuperación

UNDO (Immediate update - log -> mod -> commit)

- Al modificar un ítem se guarda el log en disco (v_{old})
- El registro (log) debe ir a disco antes que la actualización (WAL)
- La modificación debe guardarse en disco antes de hacer commit
- Se escribe el commit en el log, en disco (FLC)

Al recuperar:

- Se recorre el log de adelante hacia atrás
- deshaciendo las modificaciones de Tx sin commitear
- Se agrega al log un ABORT de las Tx sin commitear (volcado a disco)

REDO (Deferred update - log -> commit -> mod)

- Al modificar un ítem se guarda el log en disco (v_{new})
- Se escribe el commit en el log, en disco (FLC)
- Se guarda la modificación en disco

Al recuperar:

- Se analiza qué Tx commitearon
- Se recorre el log desde el principio, rehaciendo los cambios de Tx commiteadas
- Se agrega un abort de las Tx no commiteadas (volcado a disco)

UNDO-REDO (Deferred update - log -> mod/commit)

- Al modificar un ítem se guarda el log en disco (v_{old}, v_{new})
- El registro (log) debe ir a disco antes que la actualización (WAL)
- Se escribe el commit en el log, en disco (FLC)
- La modificación se puede volcar antes o después del commit

Al recuperar:

- Se recorre hacia atrás, deshaciendo las mod de Tx sin commit
- Se recorre hacia adelante, rehaciendo las mod de Tx con commit
- Se agrega un abort de las Tx no commiteadas (volcado a disco)

Puntos de Control (Checkpoints)

Indica que los ítems modificados hasta ese punto han sido almacenados en disco.

Checkpoints Inactivos: Pausa las Tx para volcar todos los buffers a disco.

- **UNDO:**

- Dejar de aceptar nuevas Tx
- Esperar a que commiten las Tx activas
- Escribir CKPT en el log (volcar a disco)
- > Al recuperar: solo hay que ir hasta el CKPT

Checkpoints Activos: Reduce la pérdida de tiempo de ejecución

- **UNDO:**

- Escribir (BEGIN CKPT, {Tx activas})
- Esperar a que commiten las Tx activas
- Escribir (END CKPT) en el log (volcar a disco)
- > Al recuperar: Si encontramos un END, solo hay que ir hasta el BEGIN
- > Si no: Solo hasta la Tx más antigua del BEGIN

- **REDO:**

- Escribir (BEGIN CKPT, {Tx activas}) (volcar a disco)
- Volcar a disco todas las modificaciones de Tx commiteadas
- Escribir (END CKPT) en el log (volcar a disco)
- > Al recuperar: Si encontramos un END, rehacer hasta la Tx más antigua del BEGIN
- > Si no: Un BEGIN solo no sirve, buscar un checkpoint anterior

- **UNDO-REDO**

- Escribir (BEGIN CKPT, {Tx activas}) (volcar a disco)
- Volcar a disco las modificaciones previas al BEGIN CKPT
- Escribir (END CKPT) en el log (volcar a disco)
- > Al recuperar: Deshacer la Tx más antigua del BEGIN que no haya commiteado
- > Rehacer operaciones posteriores al BEGIN que hayan commiteado

Procesamiento de consultas/Costos

Informacion

- $n(R)$: Cantidad de Tuplas
- $B(R)$: Cantidad de Bloques
- $F(R)$: Factor de Bloque = n/B
- $V(A,R)$: Variabilidad. Cantidad de valores distintos de A en R
- $H(I(A,R))$: Height. Altura del Índice I por el atributo A en R
- M: Cantidad de bloques de memoria
- Índices: Primario, Clustering, Secundario

Selección

Busqueda Lineal

Costo = B

Primary Index Scan

Costo Tree = $H+1$

Costo Hash = 1

Clustering Index Scan

Costo = $H + \text{ceil}(B / V)$

Secondary Index Scan

Costo = $H + \text{ceil}(n / V)$

Cantidad de Tuplas

Cantidad = n/V

Proyección

X es superclave (no elimina duplicados) o **sin DISTINCT**

Costo = B

X no es superclave - $B < M$: sort en memoria

Costo = $2B \text{ceil}(\log_M(B)) - B$

X no es superclave - $B > M$: sort externo

Costo = $B(R) + 2B(\pi R)$

Junta

Loops Anidados

Costo = $B(R) + \text{ceil}(B(R)/(M-2)) * B(S)$

Loop Unico

Costo = $B(R) + n(R) * \text{costo_idx_scan}(S)$

> Requiere un índice por el atributo de junta

Sort Merge

Costo en Memoria = $B(R) + B(S)$

Costo Externo =

$B(R) + B(S) + 2 B(R) \text{ceil_log}_{M-1}(B(R)) + 2 B(S) \text{ceil_log}_{M-1}(B(S))$

Junta Hash Grace

Costo = $3 * (B(R) + B(S))$

> $\min(B(R), B(S))/M \leq K \leq \min(M, V(R), V(S))$

> Izq: restricciones para la 2da etapa. Der: Restricciones para el hasheado.

> $K \approx \sqrt{\min B}$

Cantidad de Tuplas

Cantidad = $n(R) n(S) / \max(V(R), V(S))$

Cantidad junta por FK = $\max(n(R), n(S))$

$F(R * S) = 1/[1/F(R) + 1/F(S)]$

$B(R * S) = B(R) * B(S) * [F(R) + F(S)] / \max V$

Operadores de conjuntos

$\text{cost}(R [U | \cap | -] S) = \text{cost}(\text{OrdM}(R)) + \text{cost}(\text{OrdM}(S)) + 2 \cdot B(R) + 2 \cdot B(S)$

Pipelining

- Tendremos varios operadores en el árbol de consultas
- El resultado de un operador será la entrada de otro operador

Seguridad

- **Confidencialidad:** No ofrecer la información a individuos no autorizados.
- **Integridad:** Asegurar su correctitud durante el ciclo de vida.
- **Disponibilidad:** Asegurar que la información esté disponible cuando es requerida por personas autorizadas.
- **No repudio:** Que nadie que haya accedido a cierta información pueda negar haberlo hecho.

RBAC: Control de acceso basado en roles. Permisos por rol (Escritura / Lectura de una tabla) e incluso políticas por fila (postgre)

- **Criterio del menor privilegio posible:** si un usuario no va a realizar una determinada operación, entonces no debe tener permisos para realizarla
- **División de responsabilidades:** Para completar una tarea sensible debería ser necesaria la participación de roles mutuamente excluyentes, cada uno realizando una operación distinta.
- **Abstracción de datos:** Los permisos son abstractos. Las operaciones permitidas/prohibidas dependen de las propiedades del objeto en cuestión.

Data Warehousing

OLTP: On-line transaction processing

OLAP: On-line analytical processing (data analytics)

Consultas Básicas

1. Hallar los tweets del usuario con userid "818839458"

```
collection.find({"user_id": "818839458"})
```

```
{user_id: "818839458"}  
{"user.id_str": "818839458"}
```

2. Hallar aquellos tweets que tengan más de 500000 retweets.

```
collection.find({"retweet_count": {"$gt": 500000}})
```

```
{retweet_count: {$gt: 500000}}
```

3. Mostrar la cantidad de retweets de los tweets que se hayan hecho desde Argentina o Brasil. → campo "place.country"

```
collection.find(  
    {$or:[  
        {"place.country": "Argentina"},  
        {"place.country": "Brasil"}  
    ]},  
    {"retweet_count": 1}  
)
```

```
    {$or: [  
        {"place.country": "Argentina"},  
        {"place.country": "Brasil"},  
    ]}
```

```
{"place.country": {$in: ["Argentina", "Brasil"]}}  
{"place.country": /Argentina|Brasil/}
```

4. Hallar los usuarios que Consultas Básicas

1. Hallar los tweets del usuario con userid "818839458"

```
collection.find({"user_id": "818839458"})
```

```
{user_id: "818839458"}
```

```
{"user.id_str": "818839458"}
```

2. Hallar aquellos tweets que tengan más de 500000 retweets.

```
collection.find({"retweet_count": {"$gt": 500000}})
```

```
{retweet_count: {$gt: 500000}}
```

3. Mostrar la cantidad de retweets de los tweets que se hayan hecho desde Argentina o Brasil. → campo "place.country"

collection.find(

{\$or:[

{"place.country": "Argentina"},

{"place.country": "Brasil"}

]}},

{"retweet_count": 1}

)

{\$or: [

```
{"place.country": "Argentina"},
```

```
{"place.country": "Brasil"},
```

```
}]}
```

```
{"place.country": {$in: ["Argentina", "Brasil"]}}
```

```
{"place.country": /Argentina|Brasil/}
```

4. Hallar los usuarios que tengan tweets con 200000 o más retweets y sean en idioma español. → campo "lang"

```
{
    lang: 'es',
    retweet_count: {$gte: 200000}
}
collection.find(
    {"retweetConsultas Básicas
```

1. Hallar los tweets del usuario con userid "818839458"

collection.find({“user_id”: “818839458”})

```
{user_id: "818839458"}
```

```
{"user.id_str": "818839458"}
```

2. Hallar aquellos tweets que tengan más de 500000 retweets.

collection.find({"retweet_count": {"\$gt": 500000}})

```
{retweet_count: {$gt: 500000}}
```

3. Mostrar la cantidad de retweets de los tweets que se hayan hecho desde Argentina o Brasil. → campo "place.country"

```
collection.find(
```

```
 {$or:[
```

```
 {"place.country": "Argentina"},
```

```
 {"place.country": "Brasil"}
```

```
 ],
```

```
 {"retweet_count": 1}
```

```
)
```

```
 {$or: [
```

```
 {"place.country": "Argentina"},
```

```
{"place.country": "Brasil"},  
}  
  
{"place.country": {$in: ["Argentina", "Brasil"]}}  
  
{"place.country": /Argentina|Brasil/}
```

4. Hallar los usuarios que tengan tweets con 200000 o más retweets y sean en idioma español. → campo "lang"

```
{  
    lang: 'es',  
    retweet_count: {$gte: 200000}  
}  
  
collection.find(  
  
    {"retweet tengan tweets con 200000 o más retweets y sean en idioma español. → campo  
     "lang"  
    {  
        lang: 'es',  
        retweet_count: {$gte: 200000}  
    }  
    collection.find(  
        {"retweet_count": {$gte: 200000}, "lang": "es"},  
        {"user": 1}  
    )
```

5. Mostrar la cantidad de retweets para los tweets que no se hayan hecho en Argentina ni Brasil, pero sí tengan un lugar definido y sean en español.

```
collection.find(
```

```

{
    "$and": [
        { "place.country": {$nin: ["Argentina", "Brasil"]} },
        { "place.country": {$ne: null} },
        { "lang": "es" }
    ]
},
{
    "retweet_count": 1
}
)

collection.find(
{
    "$and": [
        { "place.country": {$nin: ["Argentina", "Brasil", null]} },
        { "lang": "es" }
    ]
},
{
    "retweet_count": 1
}
)

collection.find(
{
    "$and": [
        { "place.country": {$nin: ["Argentina", "Brasil"]} },
        {"place.country": {$exists: true}}
        { "lang": "es" }
    ]
},
{
    "retweet_count": 1
}
)

```

```
{
  'place.country': {
    '$nin': [
      'Argentina', 'Brasil', null
    ]
  },
  'lang': 'es'
}
```

PARA LAUT!!!!!

```
{$and: [
  {lang: 'es'},
  {place: {$exists: true}},
  {"place.country": {$nin: [
    "Argentina", "Brasil"
  ]}},
  {"place.country": {$exists: true}}
],
{retweet_count: 1, "place.country": 1, lang: 1}
```

6. Mostrar los screen name de aquellos usuarios que tengan "Juan" como parte de su nombre.

```
collection.find(
  {"user.name": /juan/i},
  {"user.screen_name": 1}
)

{
  "user.screen_name": /Juan/
}, {screen_name: "$user.screen_name"}
```

7. Mostrar de los 10 tweets con más retweets, su usuario y la cantidad de retweets.

```
collection.find({}, {"retweet_count": 1, "user": 1}).sort("retweet_count": -1).limit(10)
```

```
{},  
{retweet_count: 1, "username": "$user.screen_name"},  
{retweet_count: -1},  
{limit: 10}
```

Consultas de Agregación

1. Mostrar de los 10 tweets con más retweets, su usuario y la cantidad de retweets. Ordenar la salida de forma ascendente.

```
collection.aggregate([  
    {$project: {retweet_count: 1, user: 1}},  
    {$sort: {retweet_count: -1}},  
    {$limit: 10},  
    {$sort: {retweet_count: 1}}  
])
```

```
[  
{  
    $project:  
    {  
        retweet_count: 1,  
        user: 1  
    }  
},  
{  
    $sort:  
    {
```

```
    retweet_count: -1
  }
},
{
  $limit:
    10
},
{
  $sort:
    {
      retweet_count: 1
    }
}
]

[
{
  $sort:
    {
      retweet_count: -1
    }
},
{
  $limit:
    10
},
{
  $project:
    {
      user_scream_name: "$user.screen_name",
      retweet_count: 1
    }
}
]
```

2. Encontrar los 10 hashtags más usados.

```
[  
  {  
    $unwind:  
      {  
        path: "$entities.hashtags",  
        preserveNullAndEmptyArrays: false  
      }  
    },  
    {  
      $group:  
        {  
          _id: "$entities.hashtags.text",  
          cantUsos: {  
            $count: {}  
          }  
        }  
    },  
    {  
      $sort:  
        {  
          cantUsos: -1  
        }  
    },  
    {  
      $limit: 10  
    }  
]
```

3. Encontrar a los 5 usuarios más mencionados. (les hicieron @)

```
[  
  {  
    $match: {
```

```
        "entities.user_mentions": { $ne: null },
        "entities.user_mentions": { $ne: [] }
    }
},
{
    $unwind: "$entities.user_mentions"
},
{
    $group: {
        _id: "$entities.user_mentions.screen_name",
        cantidad: { $count: {} }
    }
},
{
    $sort: { cantidad: -1 }
},
{
    $limit: 5
}
})
```

4. Hallar la cantidad de retweets promedio para los tweets que se hayan hecho desde Argentina y aquellos que no.

```
[  
  { $group:{  
    _id:{  
      deArgentina: {  
        $cond: [  
          { $eq: ["$place.country",  
"Argentina"] }, true, false  
        ]  
      }  
    },  
    promRetweet: { $avg: "$retweet_count" }  
  }]
```

```
},
{
    $project: {
        _id: 0,
        deArgentina: "$_id.deArgentina",
        promRetweet: 1
    }
}
]
```

5. Por cada usuario obtener una lista de ids de tweets y el largo de la misma.

```
[
{
    '$group': {
        '_id': '$user',
        'lista_de_tweets': {
            '$push': '$_id'
        },
        }
    },
    {
        '$addFields': {
            'largo_lista': {
                '$size': 'lista_de_tweets'
            }
        }
    }
]
```

```
[
```

```
{
```

```
'$group': {
    '_id': '$user',
    'lista_de_tweets': {
        '$push': '$_id'
    },
    'contados': {
        '$count': { }
    }
},
]
}
```

6. Hallar la máxima cantidad de retweets totales que tuvo algún usuario.

```
db.tweets.aggregate([
    { $group: {
        _id: "$user",
        cantRT: { $sum: "$retweet_count" }
    }
},
{ $sort: { cantRt: -1 }
},
{ $limit: 1
}
])
```

7. Hallar para cada intervalo de una hora cuántos tweets realizó cada usuario.

```
[  
  {  
    '$project': {  
      'user': 1,  
      'hora': {  
        '$hour': '$created_at'  
      }  
    }  
  }, {  
    '$group': {  
      '_id': {  
        'user_name': '$user.name',  
        'hora': '$hora'  
      },  
      'cantidad': {  
        '$count': {}  
      }  
    }  
  }, {  
    '$group': {  
      '_id': '${_id.user_name}',  
      'horas_tweets': {  
        '$push': {  
          'hora': '${_id.hora}',  
          'cantidad': '$cantidad'  
        }  
      }  
    }  
  }  
]
```

https://docs.google.com/document/d/1nMS0fLQXf-YR2qaKwa_JOw3Zgc-Glvmj0emTp-Mm4jk/edit?usp=sharing

1. Muestre en orden alfabetico, los nombres de las primeras 10 personas apellidoadas 'Smith'.

```
MATCH (n:Person) WHERE n.surname = "Smith" return n.name order by n.name asc limit 10 para vas re rapido
```

```
MATCH (n:Person)
WHERE n.surname = "Smith"
RETURN n.name, n.surname
ORDER BY n.name ASC
LIMIT 10
```

```
MATCH (ce:Person {surname: "Smith"})
RETURN ce.name
ORDER BY ce.name
LIMIT 10
```

2. Muestre la marca y modelos de los vehículos de año 2013.

```
match (v:Vehicle{year: '2013'}) return v.make,v.model
```

```
MATCH(v:Vehicle) WHERE v.year = '2013' RETURN v.make, v.model
```

v.make	v.model
1 "Chevrolet"	"Tahoe"
2 "Nissan"	"Altima"
3 "Volvo"	"C70"
4 "BMW"	"X6"

```
MATCH(v:Vehicle{year:"2013"})
RETURN v.make, v.model
```

3. Muestre el nombre, apellido y rango de los oficiales cuyos apellidos comiencen con 'Mc', ordenados por rango (rank).

```
MATCH (o:Officer) WHERE o.surname contains "Mc" RETURN o.name, o.surname, o.rank ORDER BY o.rank DESC
```

```
match(o:Officer) where o.surname contains 'Mc' return o.name,o.surname, o.rank  
order by o.rank
```

(son 24 valores)

n.name	n.surname	n.rank
"Kinna"	"McHan"	"Chief Inspector"
"Gwyn"	"McGaughie"	"Chief Inspector"
"Jessie"	"McGavin"	"Inspector"
"Jeannie"	"McCord"	"Inspector"
"Hugibert"	"Mc Gee"	"Inspector"

4. Muestre el grafo de las locations en el área M30. Cuantos nodos hay?
210

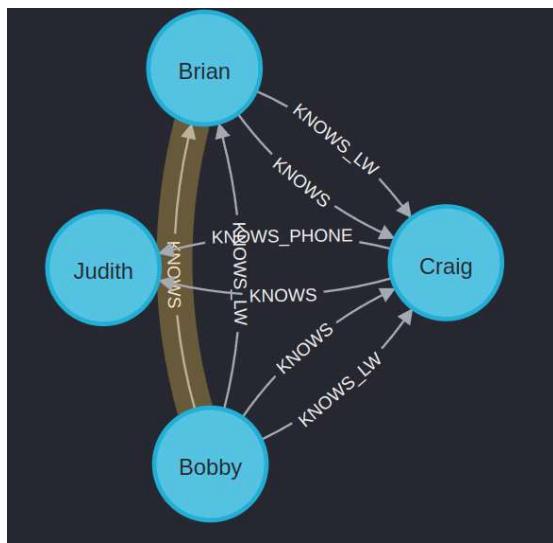
```
MATCH (l:Location) WHERE l.postcode contains "M30" RETURN l
```

```
MATCH (l:Location) -[:LOCATION_IN_AREA]-> (a:Area {areaCode: 'M30'})  
RETURN COUNT(l)
```

```
MATCH (a:Area)-[d:LOCATION_IN_AREA]-(l:Location)  
WHERE a.areaCode="M30"  
RETURN a,l
```

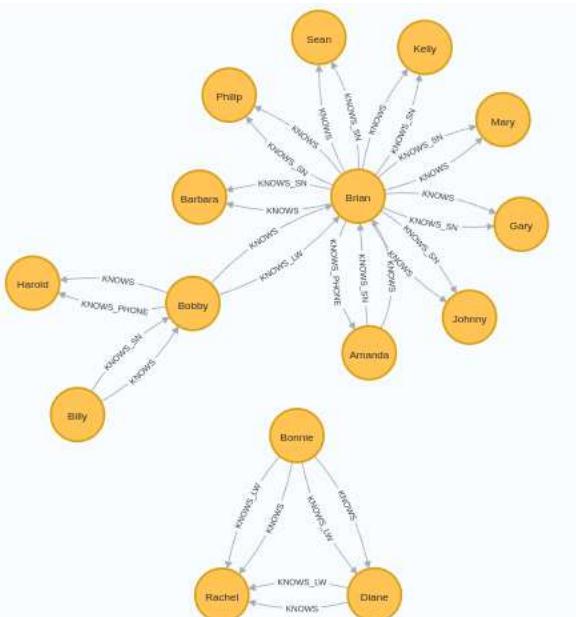
5. Muestre el grafo de todos que conocen a alguien que conoce a Gordon Craig.

```
match (p:Person)-[*1]-(gc:Person{surname: 'Gordon', name: 'Craig'}) return p
```



```
MATCH (n:Person)-[:KNOWS]-(a:Person)-[:KNOWS]-(p:Person{surname: 'Gordon', name: 'Craig'})
```

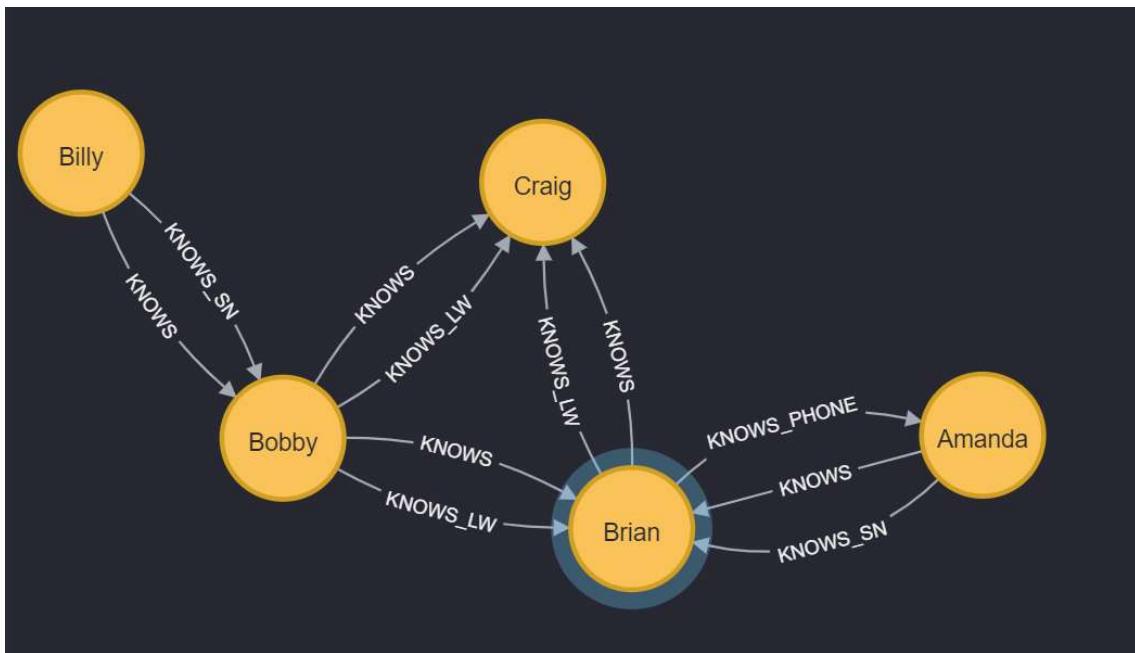
```
RETURN n
```



```
MATCH (o:Person)-[e:KNOWS]->(n:Person)-[d:KNOWS]->(m:Person)
```

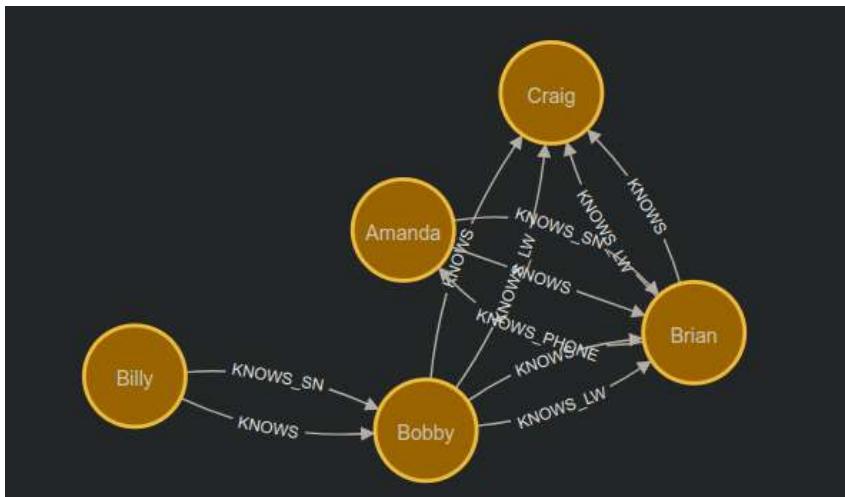
```
WHERE m.surname = "Gordon" AND m.name="Craig"
```

```
RETURN o,n,m
```



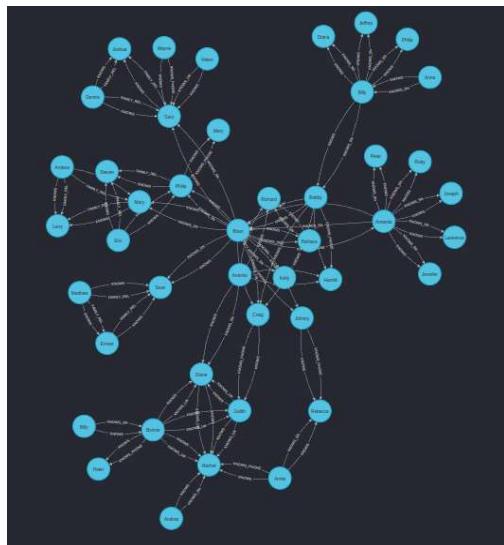
```
MATCH (p1:Person) -[:KNOWS]-> (p2:Person) -[:KNOWS]-> (p3:Person {surname: "Gordon", name: "Craig"})
```

RETURN p_1, p_2, p_3



6. Muestre las personas que están a distancia 3 de Gordon Craig.

```
match (p:Person)-[*3]-(gc:Person{surname: 'Gordon',name: 'Craig'}) return p
```



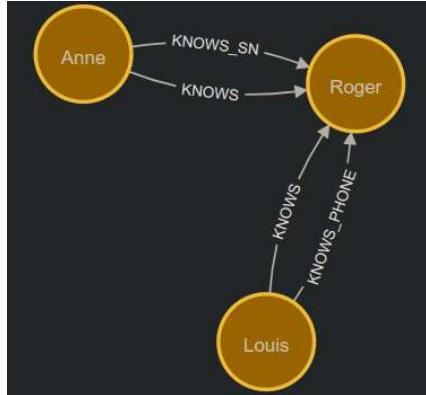
```
MATCH  (p1:Person)-[*3..3]-(p2:Person)  
WHERE  p2.name = "Craig" AND p2.surname = "Gordon"  
RETURN  p1
```

7. Muestre **ningún** criestre las personas conocidas de Roger Brooks que no participaron men.

```
match (p:Person),(c:Crime),(rb:Person{name: 'Roger',surname: 'Brooks'})  
WHERE (p)--(rb) and NOT (p)--(c) return p,rb
```

```
MATCH (p:Person)-[k:KNOWS]-(rb:Person{name: 'Roger',surname: 'Brooks'})  
WHERE NOT (p)-[:PARTY_TO]-(:Crime) return p, rb
```

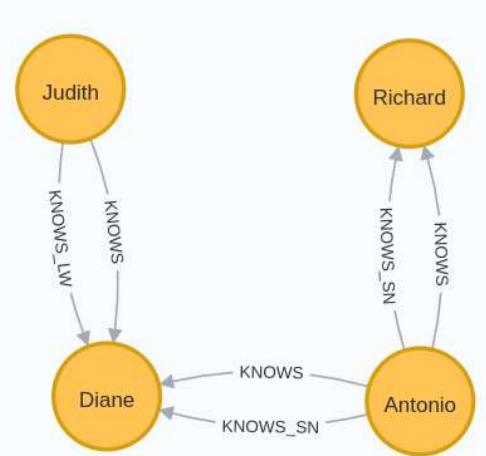
```
MATCH (p1:Person) -[:KNOWS]-> (p3:Person {surname: "Brooks", name: "Roger"})  
WHERE NOT ((p1) -- (:Crime))  
RETURN p1, p3
```



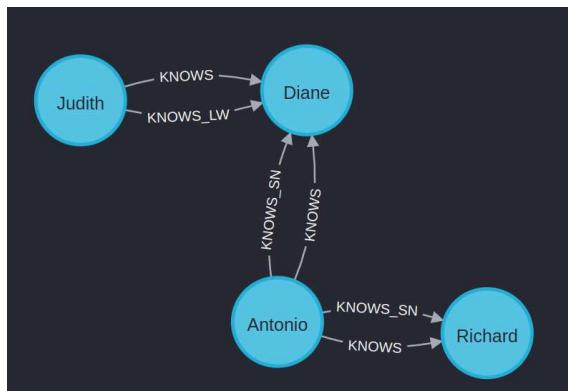
8. Muestre el camino más corto de Judith Moore a Richard Green.

```
MATCH s=shortestPath((a:Person{surname: 'Moore',name: 'Judith'})-[*]-(b:Person{surname: 'Green',name: 'Richard'}))
```

Return s



```
match s=shortestPath((jd:Person{name: 'Judith', surname: 'Moore'}))-[*]-(rg:Person{name: 'Richard', surname: 'Green'})) return s
```



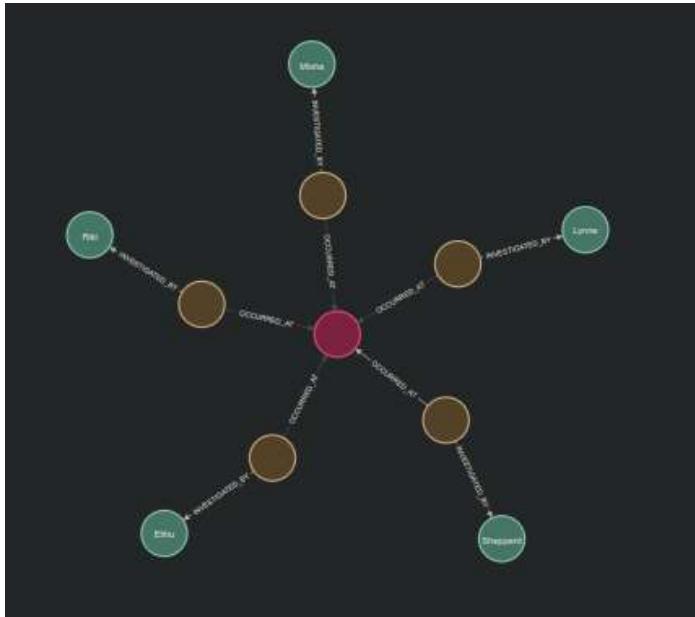
9. Encuentre los oficiales que investigaron los crímenes cometidos en 165 Laurel Street.

```
match (c:Crime),(l:Location{address: '165 Laurel Street'}),(o:Officer) WHERE
(o)--(c)--(l) return o
```

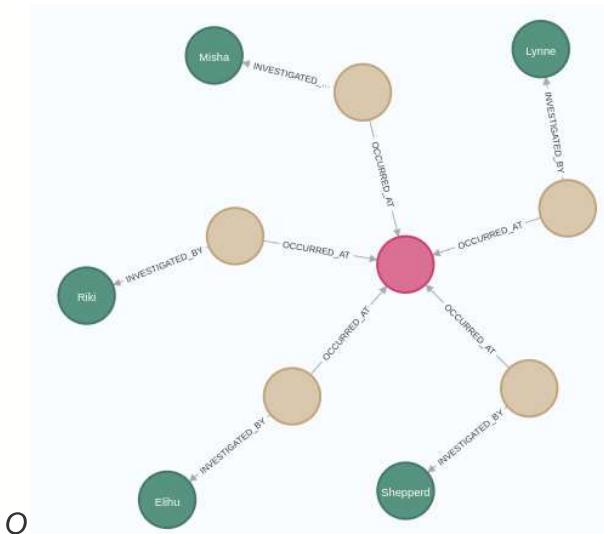


```
MATCH(o:Officer)-[:INVESTIGATED_BY]-(c:Crime)-[]-(l:Location{address: '165 Laurel Street'}) RETURN o
```

```
MATCH (o:Officer) -[]- (c:Crime) -[]- (l:Location {address: '165 Laurel Street'})  
RETURN o, c, l
```



```
MATCH  
(o:Officer)<-[:INVESTIGATED_BY]-(c:Crime)-[:OCCURRED_AT]-(l:Location{address : "165 Laurel Street"})  
RETURN o,c,l
```



10. Obtenga el modelo, marca y año del auto más viejo de la base.

`MATCH (a:Vehicle) with min(a.year) as minYear match (v:Vehicle) where v.year = minYear return v`

```

MATCH (v:Vehicle)
WITH MIN(v.year) as masviejo
MATCH (v1:Vehicle)
WHERE v1.year = masviejo
RETURN v1.make, v1.model, v1.year
  
```

v1.make	v1.model	v1.year
"Chrysler"	"Imperial"	"1926"

`match (v:Vehicle) return v.model, v.make, v.year order by v.year limit 1`

```

MATCH (a:Vehicle)
WITH MIN(a.year) AS maxyear
MATCH (v:Vehicle)
WHERE v.year = maxyear
RETURN v.make, v.model, v.year

```

v.make	v.model	v.year
"Chrysler"	"Imperial"	"1926"

11. ¿A qué distancia se encuentra el auto más viejo de Roger Brooks?

```

match (v:Vehicle) with MIN(v.year) as oldest
match (v:Vehicle),(p:Person{name: 'Roger',surname: 'Brooks'}) where v.year = oldest
match s=shortestPath((v)-[*]-(p))
return length(s)

```

8

```

MATCH (v:Vehicle)
WITH MIN(v.year) as minyear
MATCH (v1:Vehicle)
WHERE (v1.year) = minyear
WITH v1 as oldestvehicle
MATCH s=shortestPath((p:Person {name: 'Roger', surname :'Brooks'}) -[*]-
(oldestvehicle))
RETURN LENGTH(s)

```

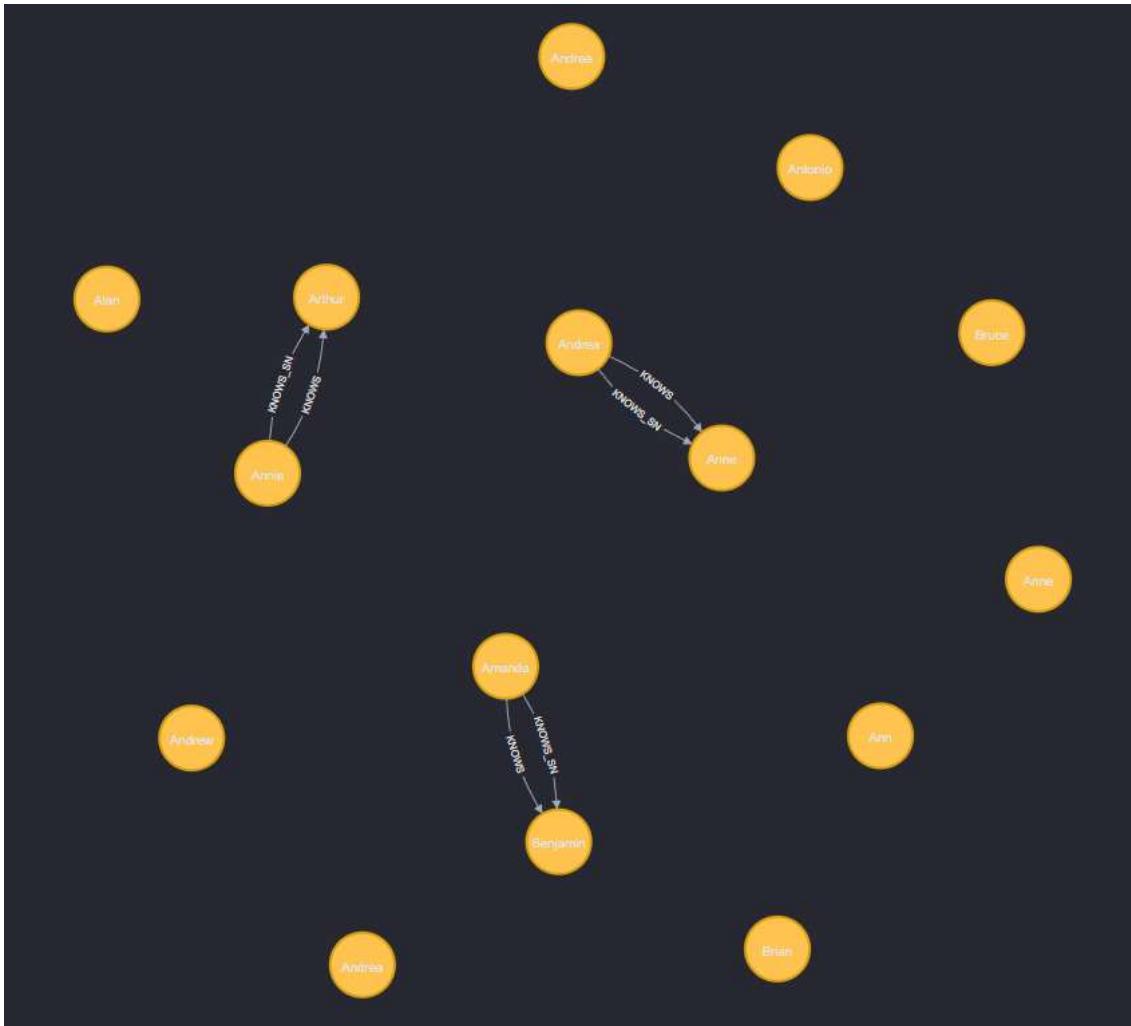
12. Devuelva el nombre y apellido de personas que conozcan más de 10 personas.

```

MATCH (p:Person) -[k:KNOWS]- (p1:Person)
WITH COUNT(*) AS cantidad_conocidos, p.name AS nombre, p.surname AS
surname
MATCH (p:Person)
WHERE cantidad_conocidos >= 10 and p.name = nombre and p.surname =
surname
RETURN p.name, p.surname, cantidad_conocidos

```

<i>p.name</i>	<i>p.surname</i>	<i>cantidad_conocidos</i>
"Benjamin"	"Hamilton"	10
"Brian"	"Austin"	10
"Ann"	"Fox"	10
"Anne"	"Rice"	11
"Anne"	"Rice"	11
"Bruce"	"Baker"	11
"Antonio"	"Hernandez"	10
"Andrea"	"George"	19
"Andrea"	"George"	19
"Arthur"	"Willis"	10
"Alan"	"Hicks"	10
"Annie"	"Duncan"	12
"Andrew"	"Foster"	11
"Amanda"	"Alexander"	13
"Andrea"	"Phillips"	10



13. Cuantas personas hay en la base? Cuantos tiene teléfono? Cuantos mail?

```
MATCH (p:Person)
```

```
RETURN COUNT(p)
```

O_o a. Bueno <3

369

```
MATCH (n:Person)
```

```
WITH COUNT(*) AS cant_pers
```

```
MATCH (m:Email)
```

```
WITH COUNT(*) AS cant_email,cant_pers
```

```
MATCH(p:Phone)
```

```
WITH COUNT(*) AS cant_tel, cant_email,cant_pers
```

```
RETURN cant_pers,cant_tel,cant_email
```

```
MATCH (p:Person)  
WITH count(p) as n1  
MATCH (p:Person)--(e:Email)  
WITH n1, count(p) as n2  
MATCH (p:Person)--(t:Phone)  
RETURN n1, n2, count(p) as n3
```

n1	n2	n3
369	328	328

```
MATCH (p:Person)  
WITH count(p) AS cantPersonas  
MATCH(p:Person)-[:HAS_PHONE]-(a:Phone)  
WITH cantPersonas, count(p) AS cantTel  
MATCH(p:Person)-[:HAS_EMAIL]-(a:Email)  
RETURN cantPersonas, cantTel, count(p)
```

cantPersonas	cantTel	count (p)
369	328	328