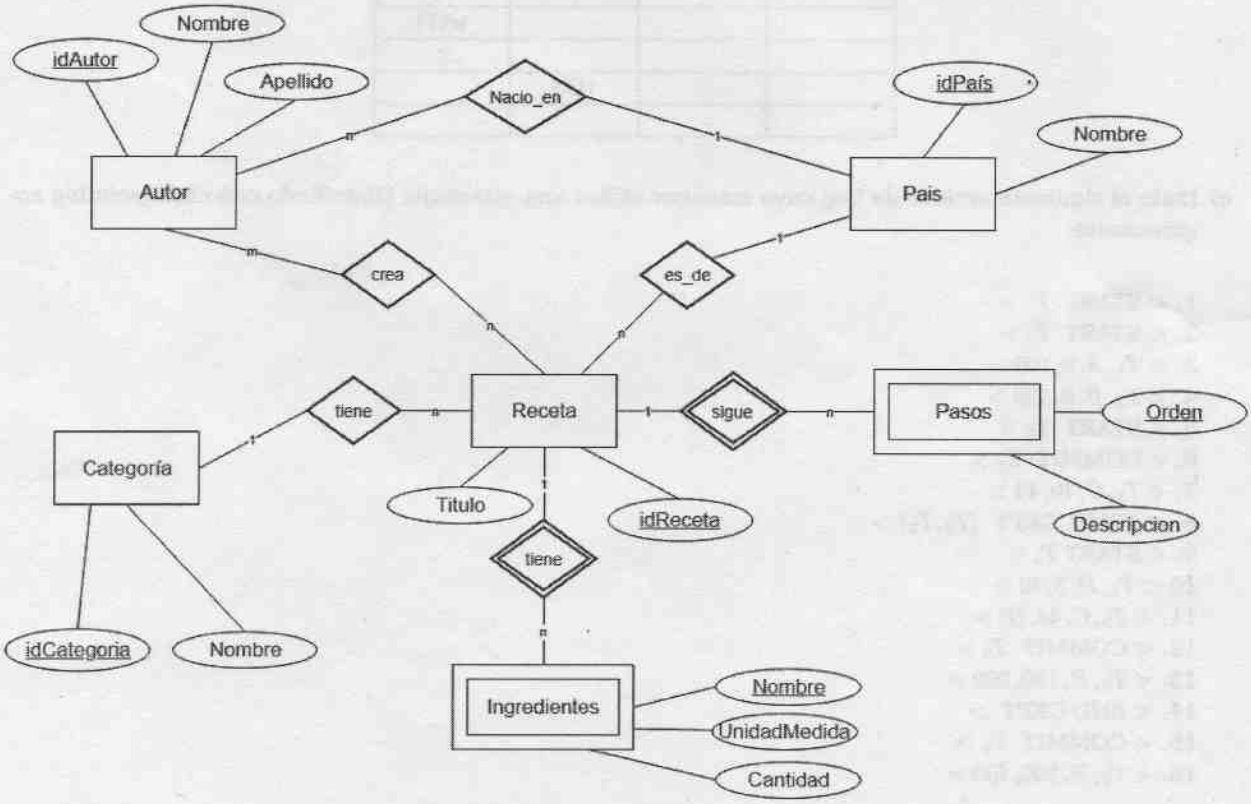
2do Parcial - 2do Cuatrimestre 2018 - Base de Datos
10 de Octubre de 2018

- Debe identificarse ***cada*** hoja con nombre, apellido, LU y su **número de orden**.
- Complete la primera hoja con la cantidad total de hojas entregadas y numere todas las hojas.
- Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Para que un ejercicio sume puntos ***no deben cometerse errores conceptuales graves***.
- La **interpretación** del enunciado forma parte de la evaluación.
- El parcial es a libro **cerrado**. Justifique sus respuestas.

Criterio de Aprobación: Se aprueba con **7**. Ejercicio 1 *5ptos*, Ejercicio 2 *5ptos*,

1 NOSQL

El DER de la figura es un modelo de datos para un sistema donde se almacenan recetas:



Se pide:

a) Document Database

(i) Dibujar el diagrama de interrelación de documentos, justificando las decisiones tomadas. Es un requisito que cuando se accede a la base de datos debe poder obtenerse para una receta inmediatamente los nombres de los autores, la categoría, los pasos y los ingredientes.

(ii) Especificar en JSON Schema el tipo de documento Receta

b) Realizar el diseño Column Family haciendo el diagrama de Chebotko para las siguientes consultas:

(i) Las recetas de los autores de un país dado ordenadas por título.

(ii) Las recetas ordenadas por categoría de la misma y dentro de ella por el título.

c) Suponga un sitio Web que permite acceder a los datos de las recetas utilizando un nombre de usuario y password. Diseñe una base de datos Clave-Valor que permita mantener el histórico de las recetas visitadas en una sesión.

2 Concurrencia y Recuperabilidad

a) Se tienen las siguientes transacciones:

$T_1 = rl_1(X); wl_1(Z); u_1(Z); u_1(X); c_1$
$T_2 = rl_2(Y); wl_2(Z); u_2(Z); u_2(Y); c_2$
$T_3 = rl_3(Z); rl_3(X); u_3(Z); u_3(X); c_3$

Se pide realizar una Historia que sea **Serializable** (pero no serial) y **No ACA**. Justificar.

b) Dada la siguiente tabla con una historia incompleta agregue al final una operación a alguna de las transacciones tal que dicha operación provoque un rollback si el planificador fuera basado en *timeStamp* **sin multiversión** pero no si fuera con **con multiversión**. **Justificar**:

T1 =100	T2 =200	T3 =300	T4 = 400
r(Y)			
	w(X)		
			r(Z)
	c		
			w(Y)
			c
		r(X)	

c) Dado el siguiente archivo de Log cuyo manager utiliza una estrategia Undo/Redo con *checkpointing no-quiescente*:

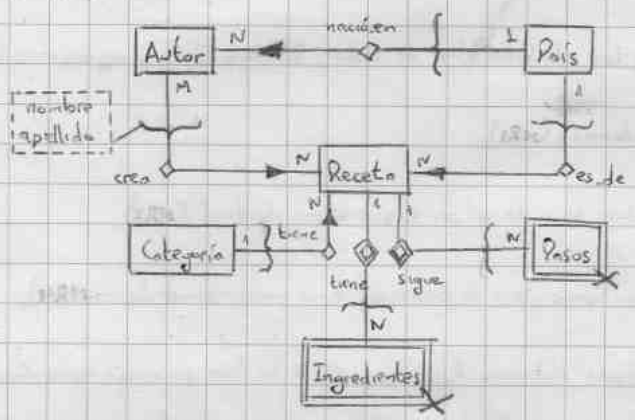
1. < START T_1 >
2. < START T_2 >
3. < $T_1, A, 0, 100$ >
4. < $T_2, B, 0, 190$ >
5. < START T_3 >
6. < COMMIT T_1 >
7. < $T_3, C, 40, 44$ >
8. < START CKPT (T_2, T_3) >
9. < START T_4 >
10. < $T_4, D, 5, 10$ >
11. < $T_3, C, 44, 50$ >
12. < COMMIT T_3 >
13. < $T_2, B, 190, 200$ >
14. < END CKPT >
15. < COMMIT T_4 >
16. < $T_2, E, 500, 400$ >

Se pide

i) Suponiendo una falla y que el último registro escrito es el 16 indicar (justificando) los pasos a seguir para la recuperación, explicando que debe hacerse con cada transacción, que modificaciones ocurren en el log si las hubiera, y los valores de los ítems modificados durante la recuperación

ii) Idem el punto i) si el último registro escrito fuera el 12.

1) a) i)



Considerando que inmediatamente al acceder a la base con la tabla de la receta debemos tener el nombre de sus autores, podemos incrustarlos a esta de forma parcial tomando los atributos que forman parte del nombre (nombre y apellido).

Para el caso de categoría podemos hacerlo (incrustación) total siendo que se desea poder obtener inmediatamente a partir de la receta. Lo mismo ocurre con los ingredientes y los pasos, aunque con la particularidad de que al ser estas entidades débiles de una interrelación con receta, no es nece Sería tener un documento aparte para representarlas así que las omitimos.

A categoría le damos una referencia a las recetas que posee para que en caso de consultar por ella se puedan buscar las recetas que tiene.

En Autor guardamos referencias tanto a las recetas que hizo (pero) al país en que nació se lo puede incrustar considerando que tiene sólo un atributo y se puede consultar directamente al acceder a la tabla de un autor.

Con País se guarda una referencia a los autores que nacieron en él así como a las recetas que son de este puesto podrían ser muchos. Siguiendo la lógica de Autor podemos insertar el País en la receta dado así se tiene directamente.

ii) Receta: PODRÉS LOS IDS ⊛ DEL AUTOR.

```
{"type": "object",
 "properties": {
    "titulo": {"type": "string"},
    "idReceta": {"type": "integer"},
    "autores": {type: "array",
       "items": {
          "type": "object",
          "properties": {
             "nombre": {type": "string"},
             "apellido": {"type": "string"}
          }
       },
    "categoría": {"type": "object",
       "properties": {
          "idCategoría": {"type": "integer"},
          "nombre": {"type": "string"}
       }
    },
    "ingredientes": {"type": array",
       "items": {
          "type": "object
          "properties": {  ⊛ LO MISMO?
             "nombre": {"type": "string"},
             "unidadMedida": {"type": "string"},
             "cantidad": {"type": "integer"},
          }
       },
       ↓
       "pasos": {"type": "array",
          "items": {
             "type": "object",
             "properties": {
                "orden": {"type": "integer"},
                "descripción": {"type": "string"}
             }
          },
       "país": {"type": "object",
       "properties": {
          "idPaís": {"type": "integer"},
          "nombre": {"type": "string"}
       }
    }
 }
```

Asumo que las unidades de medida de los ingredientes no están enumeradas (hay muchas posibles) y que orden identifica a un paso de una receta a través de un número entero.

b) i)

Recetas De Autores Por País		MR1
idPaís	K	MR2
Título	C↑	MR4
idAutor	C↑	MR5
idReceta	C↑	MR5

Siendo que queremos las recetas de los autores de un país dado tomamos los atributos clave de Autor, Receta y País, y agregamos la información restante de Receta en sus atributos al diagrama como columnas (**MR1**).

Siendo que las particiones son por país, su identificador idPaís será clave de partición por buscarse por igualdad (**MR2**).

Como necesitamos que los valores estén ordenados por Título lo tomamos como clave de clustering (asumimos ascendente → **MR4**).

Siendo que idAutor e idReceta identifican respectivamente a Autor y Receta entonces forman parte de la clave primaria como claves de clustering (**MR5**).

ii)

Recetas Ordenadas Por Categoría y Título		MR1
DUMMY	K	MR2
idCategoría	C↑	MR4
Título	C↑	MR4
idReceta	C↑	MR5

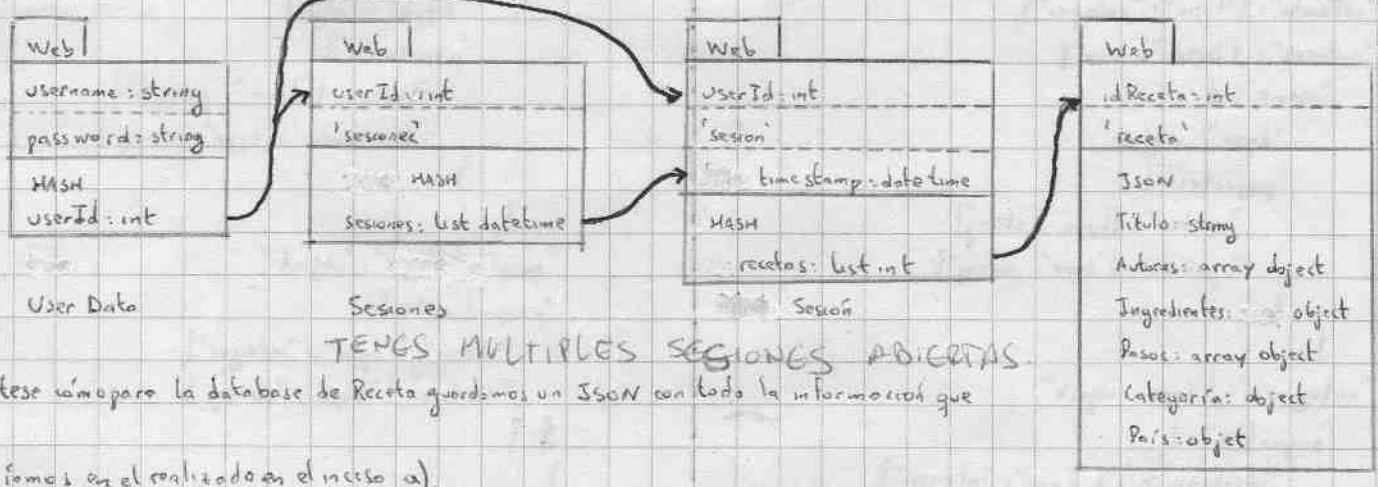
En este caso, nos piden tomar todas las posibles recetas en el diagrama y ordenarlas según Categoría y luego Título por lo que tomamos las claves de Receta y Categoría junto con los atributos de Receta (**MR1**).

Siendo que no hay atributo por igualdad y necesitamos una clave de partición agregamos una columna DUMMY (**MR2**).

El orden por Categoría viene dado por su identificador idCategoría (que asumimos de orden ascendente) y luego por Título (también ascendente) que pasan a ser claves de clustering (**MR4**).

Finalmente, como una receta se identifica por idReceta, este forma parte de las claves de clustering (**MR5**).

c)



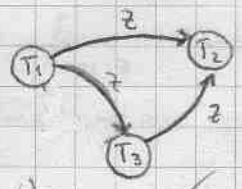
TENGS MULTIPLES SESIONES ABIERTAS

Nótese cómo para la database de Receta guardamos un JSON con toda la información que teníamos en el realizado en el inciso a).

BIEN!

2) a) $rl_1(x)$; $wl_1(z)$; $u_1(z)$; $u_1(x)$; $rl_2(y)$; $rl_3(z)$; $rl_3(x)$; $u_3(z)$; $u_3(x)$; $wl_2(z)$; $u_2(z)$; $u_2(y)$; c_1; c_3; c_2

Realizando el grafo de precedencia de esta historia tenemos:



Como no tiene ciclos, es serializable a T_1, T_3, T_2 (aplicando toposort)

~~Vemos que como para todos los conflictos se da que:~~

~~$wl_1(z) < rl_3(z) < c_1 < c_3$~~
~~$wl_1(z) < wl_2(z) < c_1 < c_2$~~
~~$rl_3(z) < wl_2(z) < c_3 < c_2$~~

~~Entonces para cada conflicto vemos que el orden de los commits respeta el de las operaciones de cada conflicto, por lo que la historia es RC,~~

~~pero como la segunda operación es al primero es ficta~~

Ahora, notando que la única instancia en que una transacción lee de otra se cuando T_3 lee z de T_1 con $wl_1(z) < rl_3(z) < c_1 < c_3$, como la lectura viene antes de que la escritura de T_1 haya sido commiteada, la historia no es ACA.

b) Siendo que se tiene la historia:

$T_1=100$	$T_2=200$	$T_3=300$	$T_4=400$
r(y)			
	w(x)		
			r(z)
	c		
			w(y)
			c
		r(x)	

Si agregamos $r(y)$ al final de la transacción T_3 vemos que en el caso que no es multiversión el planificador con timestamp se da que $TS(T_3) < WT(y)$ por lo que se produce un "read too late" que para el planificador es físicamente irrealizable y hace rollback.

Para el caso multiversión vemos en cambio que como y fue leído anteriormente por T_1 de timestamp 100 entonces se creó un versión del registro entre los timestamps 100 y 400 y de este es que T_3 puede leer para evitar rollback

c) Considerando que el checkpoint es non-quiescente en estrategia de Undo/Redo, tras la falla se debe realizar lo siguiente:

1- Recorrer desde el final del archivo log hasta el último START CKPT que tenga un respectivo END CKPT (en este caso es el del registro 8). Esto es porque con el registro esto todos los cambios de las anteriores se guardaron a disco.

~~2- Buscar en los registros posteriores transacciones incompletas y deshacer sus cambios~~ (

2- Ir hasta el START T_i más antiguo de las transacciones activas en el START CKPT anterior y de ahí en adelante buscar transacciones incompletas para deshacer sus cambios, agregando un registro ABORT T_i al final: en este caso la más antigua es T_2 con START T_2 en el registro 2 por lo que de ahí se ve que T_2 es incompleta así que se agrega <ABORT T_2> al final y se revierten

los valores de los registros de la forma: E→500; B→190; B→0 recorriendo los registros de abajo hacía arriba.

3- Partimos de START CKPT y rehacemos los cambios de los registros de todas las transacciones completas a partir de este:

Con ello vemos que rehacemos los cambios de T_4 y parte de T_3 (el registro 7 ya bajó a disco) y tenemos: D→10; C→50;

Así vemos que tenemos al final: {A=100, B=0, C=50, D=10, E=500} y el log con el agregado de <ABORT T_2> como registro 17.

ii) En este caso la sucesión de pasos es análoga sólo que como no hay un START CKPT con su respectivo END CKPT debemos ver todo el archivo log (hasta la falla en el 12). Esto hace que tengamos:

1- Comenzamos en el registro 1 → <START T_1>

2- Las transacciones incompletas son T_2 y T_4 por lo que deshacemos sus cambios y agregamos los registros <ABORT T_2> y <ABORT T_4> al final del log como registros 13 y 14. Al deshacer los cambios tenemos: D→5; B→0.

3- Rehacer los cambios de las transacciones completas a partir de este punto, las cuales son T_1 y T_3. Con esto tenemos los cambios:

A→100; C→44; C→50

De aquí vemos que al final tenemos: {A=100, B=0, C=50, D=5, E=500} y el log con el agregado de <ABORT T_2> y <ABORT T_4> como registros 13 y 14