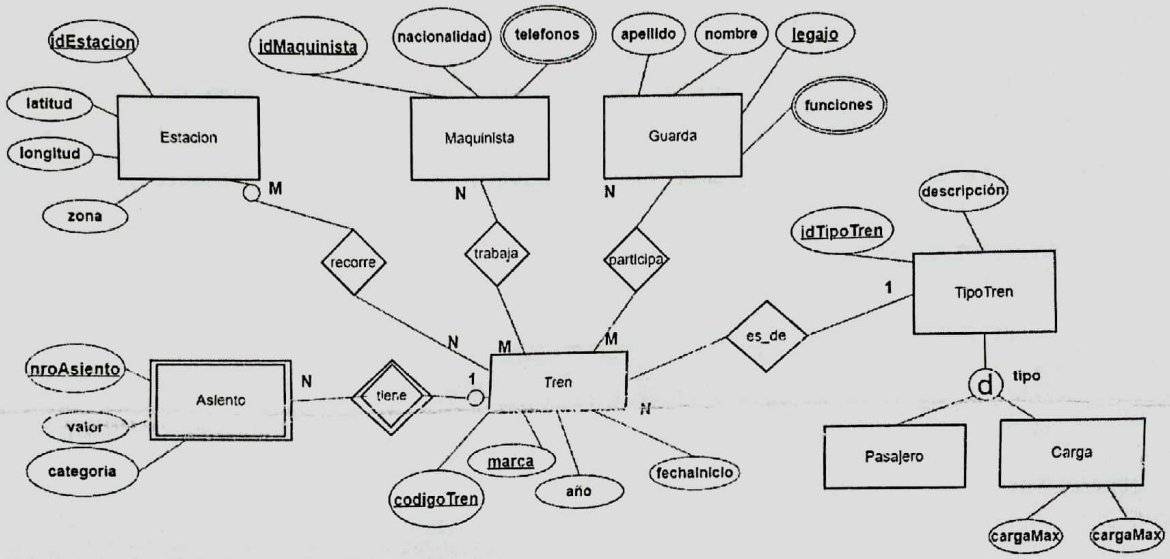
2do Parcial - Bases de Datos - 08/11/2023

- 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.

Criterio de Aprobación: Se aprueba con 7. Ejercicio 1: *5ptos*, Ejercicio 2: *5ptos*. Debe sumar al menos *3* ptos en cada ejercicio.

1. NOSQL

Dado el siguiente DER que modela los datos para una aplicación que administra un sistema de trenes.



a) Document Database: **dibujar el diagrama de interrelación de documentos**, justificando las decisiones tomadas, sabiendo que:

- Dado un TipoTren se quieren conocer todos los trenes con sus fechasInicio (que se corresponde con la fecha de inicio de actividades comerciales de dicho tren).
- El sistema requiere que cada vez que se realice una consulta por tren, se pueda obtener las estaciones que recorre, con su zona, los maquinistas con sus teléfonos y los datos asociados a los guardas.
- Dada una estación interesa también conocer los trenes que la recorren con su año

Especificar en JSON Schema del tipo de documento Estación.

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

(i) Dado un tren conocer las estaciones y sus datos, ordenados por latitud descendente.

(ii) Obtener codigotren, marca y año de los trenes de un tipo dado en los cuales trabajen maquinistas que no sean argentinos

c) Para un un sistema de gestión de la información de trenes se tienen usuarios. Estos se loguean con un userId, sin password. A partir de cada usuario se obtiene la lista de todos los trenes y, en base a ellos, se llega a los datos del tren. Con los datos de un tren se precisa saber la lista de asientos que posee, y en base a cada uno de ellos poder tener su categoría y valor en segunda instancia. También se quiere poder obtener la lista de estaciones que recorre un tren, con todos sus datos. Es importante notar, que una consulta muy frecuente es la obtención de la zona de la estación. HINT: Se puede utilizar un identificador único para el tren, idTren, que sea la concatenación del codigoTren+Marca.

2. Concurrencia y Recuperabilidad

a) Dado el siguiente schedule H en el modelo de locking ternario con soporte de update lock:

$H = wl_1(X); rl_1(Z); u_1(Z); wl_3(Z); rl_3(Y); u_1(X); wl_2(X); u_2(X); u_3(Z); rl_2(Z); wl_3(Y); u_3(Y); u_2(Z); c_1; c_2; c_3$

(i) ¿Es H legal? ¿Hay alguna transacción 2PL y si las hay cuales?

(ii) ¿Es H serializable? Si lo es: ¿Cuales son todas las historias seriales equivalentes?

(iii) ¿H es recuperable? ¿Ocurre algún dirty read en H?

Debe justificar cada respuesta correctamente utilizando las definiciones y métodos dados en las clases.

b) Dada la siguiente Historia de un planificado con timestamp:

$st_1; st_2; r_2(A); st_3; st_4; w_1(A); r_4(C); w_3(A); w_3(B); w_4(C); w_2(A); w_1(B); w_3(C)$

Suponer que t_1 escribe $A = 12$ y $B = 5$, t_2 escribe $A = 3$, t_3 escribe $A = 6, B = 11$ y $C = 2$, t_4 escribe $C = 7$ Asumir que si una transacción finaliza exitosamente, realiza commit inmediatamente después de la última operación.

(i) Decir que pasa en cada acción y qué valores quedan en A; B y C si el planificador no usa multiversion.

(ii) Decir que pasa en cada acción y qué valores se tendrán para A; B y C si el planificador usa MVTO.

c) Se tiene el siguiente log de un planificador que utiliza estrategia REDO Logging con checkpoints no-quiescente:

1. < START T_1 >
2. < $T_1, X, 12$ >
3. < START T_3 >
4. < $T_1, W, 90$ >
5. < $T_1, R, 15$ >
6. < $T_1, X, 18$ >
7. < COMMIT T_1 >
8. < START T_2 >
9. < $T_2, Y, 10$ >
10. < $T_3, M, 5$ >
11. < $T_2, C, 14$ >
12. < START T_4 >
13. < ABORT T_3 >
14. < $T_4, J, 10$ >
15. < START CKPT(T_2, T_4) >
16. < $T_2, D, 15$ >
17. < $T_4, I, 20$ >
18. < START T_6 >
19. < START T_5 >
20. < $T_6, W, 11$ >
21. < $T_5, P, 21$ >
22. < $T_5, Q, 13$ >
23. < END CKPT >
24. < COMMIT T_2 >
25. < $T_5, R, 5$ >
26. < COMMIT T_4 >
27. < COMMIT T_5 >

(i) Suponga un crash luego del paso 27. Describa detalladamente qué hace el recovery manager explicando cómo queda el log al finalizar y que sucedió con los ítems afectados por las transacciones.

(ii) Suponga un crash luego del paso 20. Describa detalladamente qué hace el recovery manager explicando cómo queda el log al finalizar y que sucedió con los ítems afectados por las transacciones.