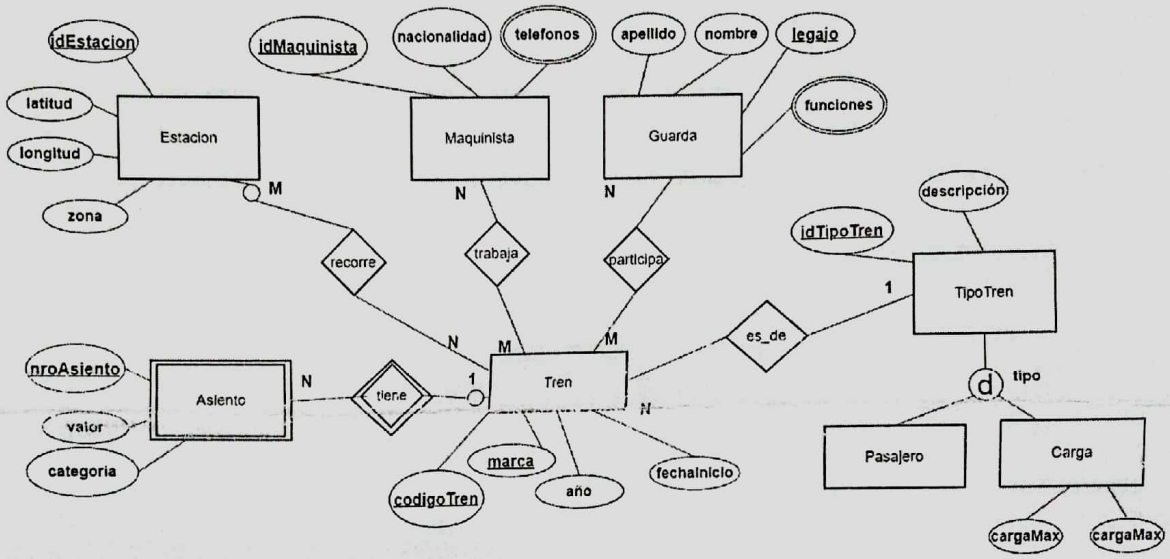
2do Parcial - Bases de Datos - 08/11/2023

- 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.

LUCIANO TARSIA

LU: 84/19 NRO. ORDEN: 25

HOJA N 1/5

FECHA 8/11

(1)

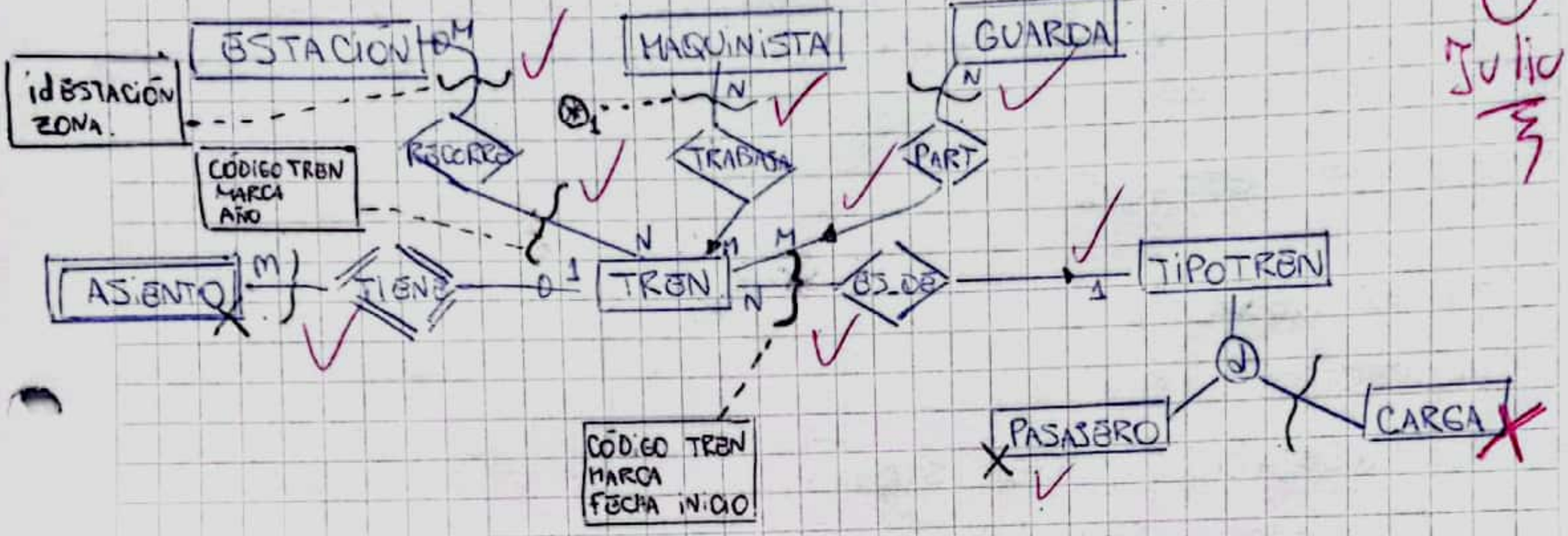
a) PARA ESTE EJERCICIO NO VOY A ESCRIBIR LOS ATRIBUTOS DE LAS ENTIDADES.

	1			2		
a	b_i	b_ii	c	a	b	c
2.9	.5	.25	1	1	1.5	2

9,15

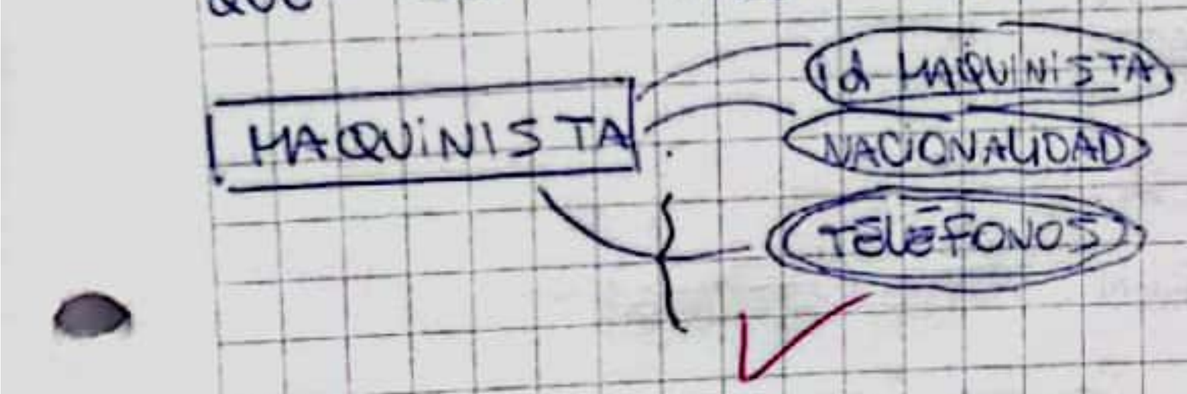
(A)

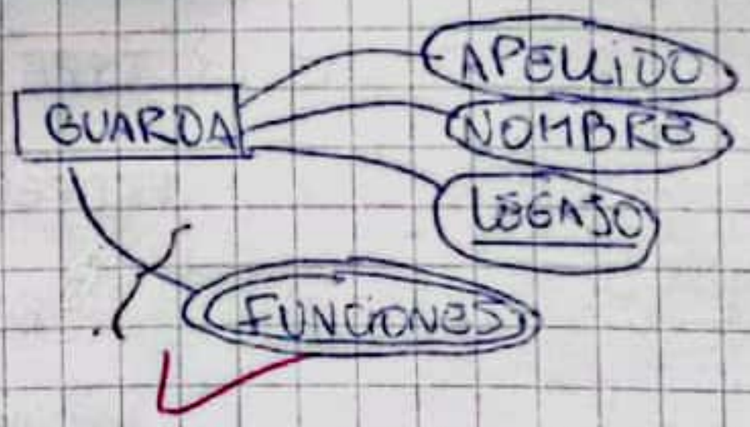
Julio



(2) id MAQUINISTA
TELEFONOS

ADEMÁS DE ESTO HAY QUE OBSERVAR QUE HAY 2 ENTIDADES CON ATRIBUTOS MULTIVALUADOS. VEAMOS QUE SE HACE CON ELLOS:





DECISIONES:

- PARA CUMPLIR CON LO PRIMERO PEDIDO EN LA CONSIGNA INCRUSTO PARCIALMENTE TREN EN TIPO TREN. LO HAGO PARCIALMENTE XQ LA CONSIGNA NO PIDE DATOS PUNTUALES.
- PARA LO SEGUNDO QUE PIDE INCRUSTO PARCIALMENTE ESTACIÓN Y MAQUINISTA EN TREN (LO INCRUSTO PARCIALMENTE XQ LA CONSIGNA PIDE DATOS PUNTUALES) E INCRUSTO GUARDIA EN TREN.
- PARA LO TERCERO INCRUSTO PARCIALMENTE TREN EN ESTACIÓN.

NOTA

ADEMÁS DE TODO ESTO ~~XXXXX~~ INCRUSTO ASIENTO EN TREN Y SUMINO ASIENTO X SER ENTIDAD DÉBIL.

TAMBIÉN SUMINO PASAJERO Y INCRUSTO CARGA EN TIPOTREN. DEBIDO A LA FLEXIBILIDAD DE LAS BASES DE DATOS DOCUMENTALES ES QUE PUEDO HACER ESTO.

POR ÚLTIMO AGREGO REFERENCIAS DONDE CONSIDERO QUE HAGA FALTA PARA NO PERDER LA RELACIÓN ENTRE ENTIDADES.

JSON DE ESTACIÓN:

```
{
  "TYPE": "OBJECT",
  "PROPERTIES": {
      "idESTACIÓN": {"TYPE": "STRING"},
      "LATITUD": {"TYPE": "STRING"},
      "LONGITUD": {"TYPE": "STRING"}
      "ZONA": {"TYPE": "STRING"}
      "TREN": {"TYPE": "ARRAY",
                "ITEMS": {"TYPE": "OBJECT",
                          "PROPERTIES": {
                              "CODIGOTREN": {"TYPE": "STRING"}
                              "MARCA": {"TYPE": "STRING"}
                              "FECHAINICIO": {"TYPE": "STRING",
                                              "FORMAT": "DATE-TIME"}
      }
  }
}
```

b) i)

(NO HAY DESIGUALDADES ⇒ NO HAY MR3)

MR1 → MR2 → MR4 (ORDEN) → MR5

Codigotron	Codigotron K	Codigotron K	Codigotron K
marca	marca K	marca K	marca K
idEstacion	idEstacion	latitud Ct	latitud Ct
latitud	latitud	idEstacion	idEstacion C↑
longitud	longitud	longitud	longitud
Zona	Zona	Zona	Zona ✓

ii)

(NO HAY ORDEN ⇒ NO HAY MR4)

MR1 → MR2 → MR3 → MR5

idTipoTron	idTipoTron K	idTipoTron K	idTipoTron K
CodigoTron	CodigoTron	nacionalidad C↑	nacionalidad C↑
marca	marca	CodigoTron	CodigoTron C↑
año	año	marca	marca C↑
idMaquinista	idMaquinista	año	idMaquinista C↑ ✓
nacionalidad	nacionalidad	idMaquinista	año

En amber Coser en MR1 selecciono las columnas.

En MR2 selecciono la clave de partición (K) (por igualdad)

En MR3 selecciono la C (por desigualdad)

En MR4 selecciono la C por el orden.

En MR5 añado las claves necesarias provenientes del "frame" del DER que estoy observando.

Hasta donde pregunté me dijeron que no hacía falta poner los tipos.

c)

TRENES

'USER_ID'

USERID: STRING

LISTA

[idTREN...]

TRENES

'idTREN'

idTREN: STRING

HASH

CODIGOTREN: STRING
MARCA: STRING
AÑO: STRING
F_INICIO: STRING
[NRO_ASIENTO...]: ARRAY

TRENES

'idTREN'

idTREN: STRING

'NRO_ASIENTO'

NRO_ASIENTO: STRING

HASH

VALOR: INT
CATEGORIA: STRING

ESTACIONES

'idESTACIÓN'

idESTACIÓN: STRING

HASH

LATITUD: STRING
LONGTUD: STRING
ZONA: STRING

ESTACIONES

'idTREN'

idTREN: STRING

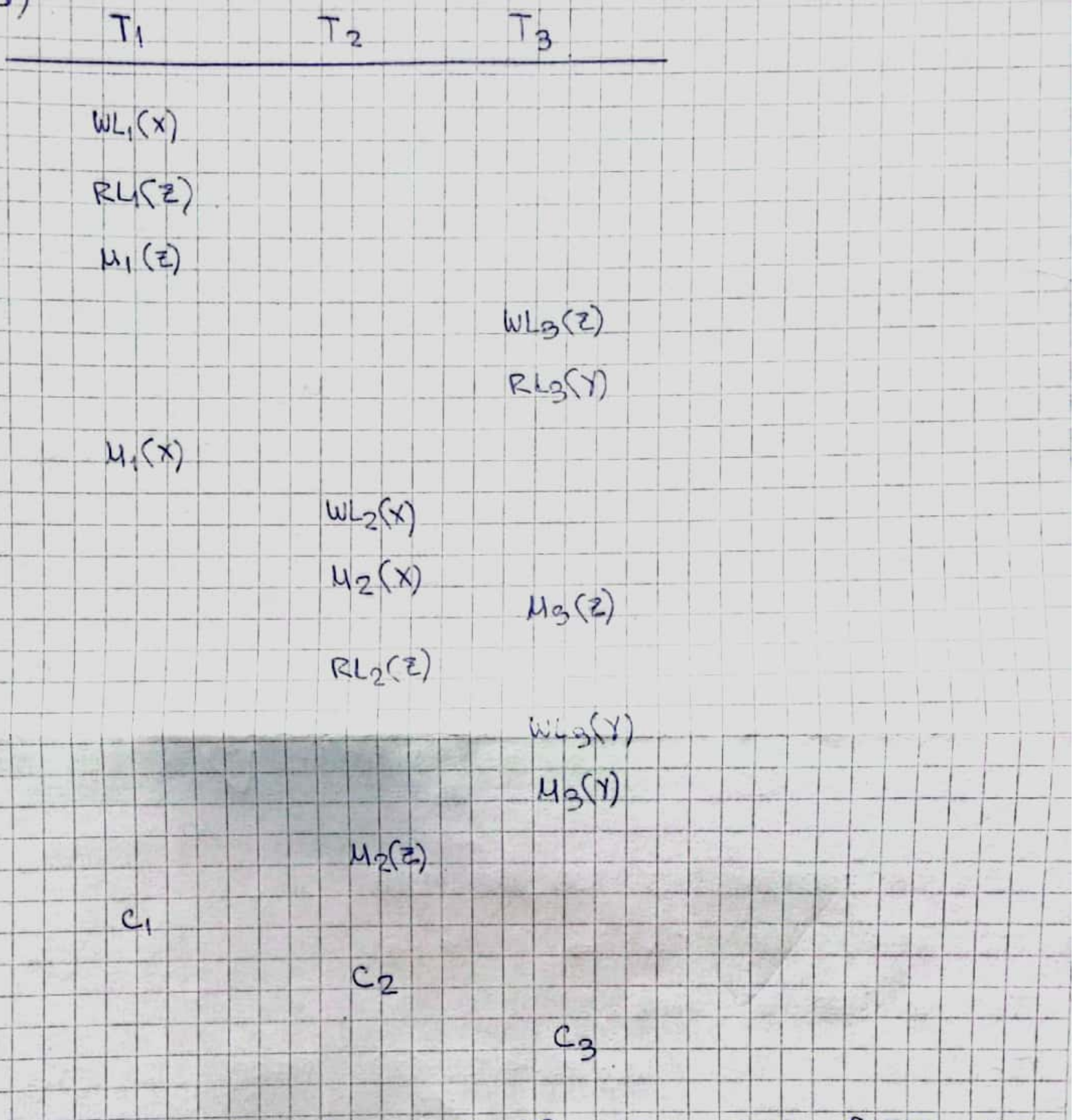
LISTA

[id_ESTACIÓN...]

LUCIANO TARSIA
LU: 84/19 N.O: 25

(2)

a)



(i) PARA QUE SEA LEGAL TIENE QUE PASAR QUE

1- Si T_i PIDE $WL_i(x)$, NO HAY UN $RL_j(x)$ O $WL_j(x)$ CON $j \neq i$ SIN SU $M_j(x)$ CORRESPONDIENTE ANTES DEL $WL_i(x)$.

2- Si T_i PIDE $RL_i(x)$, NO HAY UN $WL_j(x)$ CON $j \neq i$ SIN SU $M_j(x)$ CORRESPONDIENTE ANTES DEL $RL_i(x)$.

ESTO SE CUMPLE EN H, ASÍ QUE H ES LEGAL.

LUEGO, UNA TRASACCIÓN ES 2PL SI PRIMERO PIDE TODOS LOS LOCKS Y LUEGO LOS LIBERA. VEAMOS TRANSACCIÓN A TRANSACCIÓN SI LO CUMPLEN.

$T_1 = WL_1(x), RL_1(z), U_1(z), U_1(x), C_1$.

T_1 ES 2PL. ✓

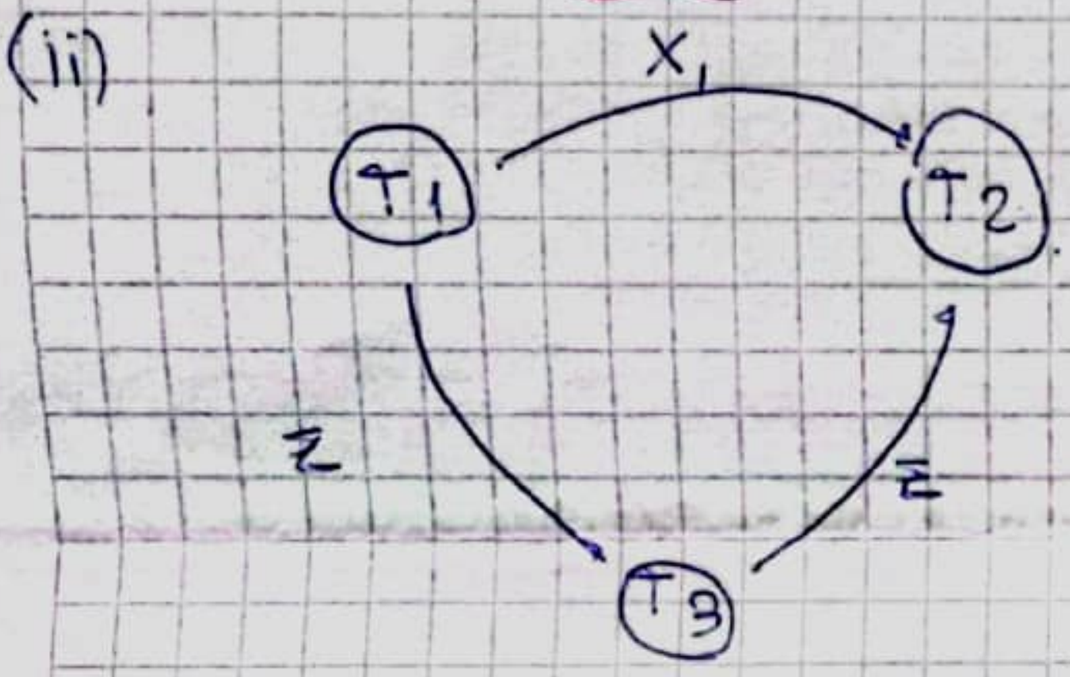
$T_2 = WL_2(x), U_2(x), RL_2(z), U_2(z), C_2$.

T_2 NO ES 2PL, ESTO SE VE EN LAS PRIMERAS 3 OPERACIONES DE LA TRANSACCIÓN.

$T_3 = WL_3(z), RL_3(y), U_3(z), WL_3(y), U_3(y), C_3$.

T3 NO ES 2PL ✓ YA QUE HACE "$U_3(z), WL_3(y)$".

(ii)



DADO QUE EL GRAFO NO TIENE CICLOS, H ES SR.

LAS HISTORIAS SR CONFLICTO EQUIVALENTE SON:

T_1, T_3, T_2 ✓

(iii) UNA HISTORIA ES RECUPERABLE sii

T_i LEE DE $T_j \Rightarrow C_j < C_i$

T_1 NO LEE DE NADIE.

T_2 LEE DE T_3 ($WL_3(z) < RL_2(z)$ PERO $C_2 < C_3$.

H NO ES RECUPERABLE.

LUCIANO TARSIA
LU: 84/19 N.O.: 25

HOJA N° 4/5
FECHA 8/11.

b)
(i) DADO EL ORDEN DE INICIO DE LAS TRANSACCIONES LE ASIGNO LOS SIGUIENTES TS.

$T_1 = 100$, $T_2 = 200$, $T_3 = 300$, $T_4 = 400$.

T_1	T_2	T_3	T_4	A	B	C.
100	200	300	400	RT=0 WT=0	RT=0 WT=0	RT=0 WT=0.

ST_1

WTL.
WT(A) < TS(T_1) < RT(A)
SE HACE ROLLBACK.

ST_2

$R_2(A)$ RT=200

ST_3

ST_4

$R_4(C)$ RT=400.

$W_1(A)$

$W_3(A)$ A=6 WT=300 C=FALSE.

$W_3(B)$ B=11 WT=300 C=FALSE.

$WT_4(C)$ C=7 WT=400 C=FALSE.

C_4 C=TRUE.

DADO QUE
WT(A) > TS(T_2)
NO SE ESCRIBE.
SE ESPERA C_3 → $W_2(A)$.

NO SE ESCRIBE → $W_3(C)$.
WT(C) > TS(T_3).

C_3 C=TRUE. C=TRUE.

C_2.

LOS VALORES QUE TERMINAN TENIENDO SON.

A = 6, ✓ B = 11 ✓, C = 7 ✓

(ii).

LO PRIMERO A NOTAR ES QUE MUTO NO VA A IMPEDIR EL WTL.

VEAMOS LA HISTORIA PASO A PASO

PRIMERO SE HACE $R_2(A)$ QUE LEE A_0.

LUEGO $W_1(A_1)$. SE HACE UN WTL POR EL $R_2(A)$. TENEMOS QUE. $TS(T_0) < TS(T_1) < TS(T_2)$

SE ABORTA T_1.

LUEGO $R_4(C)$ QUE LEE. $R_4(C_0)$.

LUEGO $W_3(A_3)$

LUEGO $W_3(B_3)$

LUEGO $W_4(C_4)$. Y T_4 HACE SU COMMIT.

LUEGO $W_2(A_2)$. Y T_2 HACE SU COMMIT.

LUEGO $W_3(C_3)$ Y T_3 HACE SU COMMIT.

FINALMENTE QUEDAN LOS SIGUIENTES VALORES.

$A_3 = 6$ ✓

$B_3 = 11$ ✓

$C_4 = 7$ ✓

$A_2 = 3$ ✓

$C_3 = 2$ ✓

LUCIANO TARSIA
LU: 84/19 NO: 25

HOJA N 5/5
FECHA 8/11

c)

(i) LO PRIMERO QUE SE HACE ES VER EL CKPT DE LA LÍNEA 15 CON SU END CKPT DE LA 23.

EL CKPT NOS GARANTIZA QUE LAS TRANSACCIONES CON COMMIT QUE ESTÁN ANTES DE LA 15 FUERON SUBIDAS A DISCO.

LUEGO, EL RECOVERY MANAGER DETECTA CUALES SON LAS TRANSACCIONES COMMITEADAS QUE FORMAN PARTE DEL STAR CKPT Y LAS CREADAS LUEGO DEL START CKPT Y HACE REDO DE ELLAS.

EN NUESTRO CASO SON: T2, T4, T5

LOS ITEMS QUEDAN:

Y=10 ✓, C=14 ✓, D=15 ✓, I=20 ✓, P=21 ✓, Q=13 ✓, R=55 ✓

ADEMÁS SE AGREGA EL ABORT DE T6 ✓ AL LOG.

(ii) EN ESTE CASO COMO PERDEMOS EL END CKPT Y NO HAY OTRO CKPT EXITOSO ANTES HAY QUE HACER REDO DE TODO.

LAS TRANSACCIONES A LAS QUE SE LE HACE REDO SON: T1, T2, T4, T6, T5 ✓

SE AGREGA EL ABORT AL LOG PARA: T2, T4, T6, T5

LOS ITEMS QUEDAN.

W=90 ✓, R=15 ✓, X=18 ✓