Resoluciones de finales y ejercicios varios de Base de Datos

Martín Buchwald
Pablo Musumeci
Florencia Bosch

Índice

1. Final 12/02/2014

E1. Dar un ejemplo de una relación $R(A, B, C, D, E)$ que satisface la dj $|x|[ABC, BDE, ACE]$ pero no satisface ninguna dmv trivial.

E2. Dado el siguiente esquema de base de datos:
P_Info(NroParte,Subparte_de, Descrip)
UsadoEn(NroParte, TipoAvion, Cant_Usa)
EnStock(NroParte, Lugar, Cantidad)
Responder la siguiente consulta en SQL: *'Para cuáles partes el total en stock es al menos la cantidad usada por un B737, y por qué monto el total excede la cantidad usada'.*
Se deberá tener en cuenta tanto la correctitud como la simplicidad.

E3. Describa en detalle el método de pipeline para la junta múltiple e indique que tipo de método de junta aplicaría en cada etapa. Justificar.

E4. Considere el siguiente plan de ejecución de las transacciones T1, T2 y T3: W1(A), R1(B), R1(C), W3(B), W1(B), W3(C), R2(C), W2(B), W2(C), R3(A)

a) Dibujar el grafo de precedencia para esta ejecución.

b) ¿Este plan es serializable? ¿Por qué? Si es serializable, dar un plan serial.

c) Expresar todos los casos donde una transacción "lee de otra".(Si T2 lee de T1, escribir T1->T2.)

1.1. Resolución

1) Pendiente charlarlo con Ale.

2. Final 05/02/2014

E1. Se tiene el esquema de relación $R(A, B, C, D, E)$ y la dependencia de junta $J = |x|[ABC, BD, CDE]$. Sabiendo $r(R)$ tiene las tuplas $< a, b, c, d, e >$, $< a', b, c', d', e'' >$ y $< a'', b', c, d', e' >$, indicar qué otra tupla debe tener también r.

E2. Sean los siguientes esquemas relacionales:
Vuelo(VueloNro, Desde, Hacia, Distancia, Partida, Arribo, Precio)
Aeronave(AId, ANombre, Rango)
Certificado(EId, AId)
Empleado(EId, ENombre, Sueldo)
La relación Empleado contiene los datos de todos los empleados de la compañía, entre ellos los pilotos. En la relación Certificado sólo figuran los pilotos certificados para volar una determinada aeronave.
Responder la siguiente consulta en SQL: *'Hallar los nombres de los empleados que sólo estén certificados para volar aeronaves de rango mayor a 2000Km, y al menos un Boeing'.*

E3. Describa detalladamente el método de junta Hash versión Grace y dar el costo del método.

E4. Dada la siguiente planificación de tareas, con valores iniciales de `salario = 1` y `tax = 2`, y sabiendo que el sistema trabaja con logging de tipo UNDO/REDO:

```
        T1                          T2                          T3
0.                                                              start
1.                                                              read tax
2.                                                              tax:= tax + 1
3.      start
4.      read salario
5.      salario := salario + 1
6.                                                              write tax
7.                                                              commit
8.                                  start
9.                                  read salario
10.                                 read tax
11.                                 tax := tax + salario
12.                                 write tax
13.                                 commit
14.-----------------------------------start checkpoint-----------------------------
15.     read tax
16.     tax := tax + 1
17.     write salario
18.-------------------------------------end checkpoint--------------------------
19.     commit
```

a) Escribir cómo serían las entradas del log, y qué línea produce cada entrada. Utilizar la nomenclatura utilizada en clase.

b) Si ocurriera un crash justo después de la línea 7, ¿cuáles transacciones habría que deshacer y cuáles rehacer?

c) Si ocurriera un crash justo después de la línea 18, ¿cuáles transacciones habría que deshacer y cuáles rehacer?

2.1. Resolución

1) Dado por la definición de una dependencia de junta, podemos decir que necesitamos que se cumpla que tenemos tres tuplas que cumplan:

$$t_1[ABC \cap BD] = t_2[ABC \cap BD] \Rightarrow t_1[B] = t_2[B]$$

$$t_2[BD \cap CDE] = t_3[BD \cap CDE] \Rightarrow t_2[D] = t_3[D]$$

$$t_1[ABC \cap CDE] = t_3[ABC \cap CDE] \Rightarrow t_1[C] = t_3[C]$$

Además, debe haber una cuarta tupla que satisfaga que:

$$t_4[ABC] = t_1[ABC]$$

$$t_4[BD] = t_2[BD]$$

$$t_4[CDE] = t_3[CDE]$$

Si nos fijamos bien, las tuplas del enunciado, en ese orden, son las tuplas t_1, t_2 y t_3, sólo necesitamos una tupla t_4:

$$t_4 =< a, b, c, d', e' >$$

2)

```
SELECT e.ENombre
FROM Empleado e
WHERE e.EId not in (SELECT c.EId
              FROM Certificado c, Aeronave a
              WHERE c.Aid = a.Aid and a.Rango <= 2000)
   and e.Eid in ( SELECT c.EId
              FROM Certificado c, Aeronave a
              WHERE c.Aid = a.Aid and a.Anombre = 'Boeing');
```

3) El método de junta Hash versión Chase se utiliza cuando ninguna de las tablas a juntar (llamémoslas R y S) entran en memoria. Lo que se busca es generar tablas que si entren en memoria (al menos una de ellas en realidad) y de manera tal que no sea necesario comparar los elementos de una partición de R con todas las particiones de S (sino, ¿dónde estaría la mejora?). Para hacer ésto, lo que se hace es generar particiones de M archivos. Se utiliza una función de hashing h ($h : A \to [0, M-1]$, donde A es el dominio de los atributos a sobre los que se hace la junta). Entonces, por cada fila de R se aplica la función de hashing para saber a cuál número de archivo se debe mandar tal fila (y luego se lo escribe, obvio). Una vez terminado, se hace lo mismo con S, pero con otros M archivos. La utilidad de todo esto es saber que si un par de filas de R y S deben juntarse, entonces necesariamente deben estar en el mismo número de archivo (que son dos archivos distintos para cada relación). Luego, para unir cada par de archivos del mismo número, se pueden aplicar distintos métodos vistos en clase, aunque se suele aplicar el método simple de Hash, que es en el cual aunque sea una de las tablas entra en memoria (suponiendo que la función de hashing anterior distribuía bien los datos, y el dominio calculado es correcto, entonces podríamos suponer que entran perfectamente en memoria). Lo que se hace es cargar tal tabla en memoria aplicando una función de hashing para almacenarla en una tabla de hash. Luego, por cada elemento de S se lo busca (por medio de sus atributos de junta) sobre la tabla de hashing, para buscar todas las filas con las que hay que hacer la junta. Finalmente, cuando se hizo cada junta parcial se unen los resultados.
Costo: como se pudo ver, para cada tabla lo que se hace es leerla una vez para luego ir creando los M archivos, lo que implica escribir todas las filas de la tabla en disco, y después se lee cada archivo otra vez para hacer la junta en sí. Por lo tanto, estamos leyendo/escribiendo a disco 3 veces por cada bloque de la tabla, entonces:

$$Costo = 3B_r + 3B_s = 3(B_r + B_s)$$

4)a) Hay que tener en cuenta a la hora de hacer este ejercicio que al modificar variables locales no es que estamos escribiendo en la base de datos. Una vez aclarado eso:

```
<START T3> | linea 0
<START T1> | linea 3
<T3, tax, 2, 3> | linea 6
```

```
<COMMIT T3> | linea 7
<START T2> | linea 8
<T2, tax, 3, 4> | linea 12
<COMMIT T2> | linea 13
<START CKPT (T1)> | linea 14
<T1, salario, 1, 2> | linea 17
<END CKPT> | linea 18
<COMMIT T1> | linea 19
```

b) Es necesario deshacer T1 (aunque en realidad no se hace nada, porque todavía no llegó a hacer nada sobre la base de datos), porque no se encuentra su commit. T3 es necesario rehacerla porque se encuentra su commit, pero no podemos estar seguro que se hayan pasado las modificaciones a disco. T2 todavía no *apareció en escena*, así que no es necesario hacer nada.

c) Es necesario deshacer T1 nuevamente, pero no es necesario rehacer T2 ni T3, pues haber encontrado el END CKPT nos asegura que todas las operaciones anteriores al START fueron flusheadas a disco, que en este caso son la totalidad de las transacciones T2 y T3 (si no hubieran checkpoints sería necesario rehacer estas transacciones).

3. Final 18/12/2013

E1. Dada la relación R(A,B,C,D,E) y las dependencias $M = \{A \twoheadrightarrow BC, B \rightarrow D, C \twoheadrightarrow E\}$. Probar usando el algoritmo de Chase que la dependencia $A \twoheadrightarrow E$ también se satisface.

E2. Sean los siguientes esquemas relacionales:
Vuelo(VueloNro, Desde, Hacia, Distancia, Partida, Arribo, Precio)
Aeronave(AId, ANombre, Rango)
Certificado(EId, AId)
Empleado(EId, ENombre, Sueldo)
La relación Empleado contiene los datos de todos los empleados de la compañía, entre ellos los pilotos. En la relación Certificado sólo figuran los pilotos certificados para volar una determinada aeronave. Responder la siguiente consulta en SQL: *'Listar los nombre y sueldo de los empleados que no son pilotos pero que su sueldo es superior al promedio de los pilotos'.*
Se deberá tener en cuenta tanto la correctitud como la simplicidad.

E3. Describir detalladamente el método de junta con índice de agrupamiento y dar su costo.

E4. Sea el siguiente log de un sistema que usa undo/redo logging. Cuál es el valor de los items A, B, C, D, E, F y G en disco después de la recuperación si la falla se produce: a) justo antes de la linea 19; b) justo antes de la línea 24; c) después de la línea 24:

```
1.  <START T1>
2.  <T1, A, 10, 50>
3.  <START T2>
4.  <T1, B, 10, 130>
5.  <T1, A, 50, 70>
6.  <T2, C, 10, 20>
7.  <T2, D, 10, 30>
8.  <COMMIT T1>
9.  <START T3>
10. <T3, E, 30, 60>
11. <T2, D, 30, 40>
12. <START CKPT(T2,T3)>
13. <T2, C, 20, 70>
14.  <COMMIT T2>
15. <START T4>
16. <T4, F, 10, 100>
17. <T4, G, 110, 10>
18. <COMMIT T3>
19. <T4, F, 100, 150>
20. <START T5>
21. <T5, C, 70, 200>
22. <END CKPT>
23. <T4, F, 150, 140>
24. <COMMIT T4>
```

3.1. Resolución

1) Teniendo $X \twoheadrightarrow Y$, entonces R se debe poder descomponer en $R_1 = XY$ y $R_2 = X(R - XY)$. Teniendo $A \twoheadrightarrow E$: Creo la descomposición $R_1 = (A, E)$ y $R_2 = (A, B, C, D)$. Utilizando el algoritmo de Chase:

	A	B	C	D	E
AE	a_1	b_{12}	b_{13}	b_{14}	a_5
ABCD	a_1	a_2	a_3	a_4	b_{25}

La dependencias multivaluada $A \twoheadrightarrow BC$ se transforma en una dependencia de junta $|x|[ABC; ADE]$. Proyectamos sobre cada caso:

ABC	A	B	C
AE	a_1	b_{12}	b_{13}
ABCD	a_1	a_2	a_3

ADE	A	D	E
AE	a_1	b_{14}	a_5
ABCD	a_1	a_4	b_{25}

Hacemos la junta de estas dos tablas:

A	B	C	D	E
a_1	b_{12}	b_{13}	b_{14}	a_5
a_1	b_{12}	b_{13}	a_4	b_{25}
a_1	a_2	a_3	b_{14}	a_5
a_1	a_2	a_3	a_4	b_{25}

Las filas 1 y 4 ya se encontraban antes, así que no las agregamos por segunda vez. Nos queda entonces:

	A	B	C	D	E
AE	a_1	b_{12}	b_{13}	b_{14}	a_5
ABCD	a_1	a_2	a_3	a_4	b_{25}
	a_1	b_{12}	b_{13}	a_4	b_{25}
	a_1	a_2	a_3	b_{14}	a_5

No es necesario aplicar la regla para la otra DMV, porque tenemos la DF $B \to D$, por lo tanto, dado que la fila 2 y 4 tienen en comun el B (a2), y distinto D, ponemos el D correcto (a_4) en la fila 4, y la fila queda completa de elementos distinguidos, por lo que la dependencia $A \twoheadrightarrow E$ también se satisface.

2)

```
SELECT e.eNombre, e.sueldo
FROM Empleado e
WHERE e.eID not in (SELECT eID FROM Certificado) and
        e.sueldo > (SELECT Avg(sueldo) FROM Empleado WHERE eID in (Select eiD From
           Certificado));
```

3) Sea la junta $R|x|S$, en el cual R tiene un atributo A, y S tiene un atributo A, para el cual S tiene un indice de agrupamiento (lo cual quiere decir que el orden lógico sobre tal índice corresponde con el orden físico). Por cada elemento de R, lo buscamos en S utilizando el índice. Esto permite acceder solo a aquellos bloques de S que contengan los elementos buscados.

Costo: Leemos cada bloque de R una única vez, y por cada elemento, lo buscamos en S. Lo que demore tal búsqueda (cada una de ellas) dependerá, en el peor de los casos, de la proporción del total de bloques de S (llegar hasta el final) sobre la imagen de valores de A en S. Tal operación se realiza por cada fila de R, por lo tanto:

$$Costo = Br + nr\frac{Bs}{V(S,A)}$$

4)a)

- A = 70
- B = 130
- C = 70
- D = 40
- E = 60
- F = 10 (no se ve el commit de T4)
- G = 110

b) Idem A, solo que no es necesario analizar T1, puesto que vemos el end del start checkpoint que no lo incluye. Ademas no se toma la modificacion de T5 sobre C porque no se ve su commit.

c)

- A = 70
- B = 130
- C = 70 (nunca se ve el commit de T5)
- D = 40
- E = 60
- F = 140
- G = 10

4. Final 24/07/2013

E1. Dada R(ABCDE) y $M = \{A \to\to BC, B \to D, C \to\to E\}$, probar usando el algoritmo de Chase que se satisface $A \to\to E$.

E2. Dada la relación R(A), usando SQL convencional escribir una consulta para hallar el valor de la mediana de A (el valor tal que la mitad de los valores son más grandes y la mitad son más chicos). Considerar que no hay duplicados de A en R, que la cantidad de elementos es impar y no se permiten nulos.

E3. Sean W(AB), X(BC), Y(CD), Z(DE) y nw=100, nx=400, ny=200, nz=300, V(W,A)=80, V(X,B)=200, V(Y,C)=200, V(Z,D)=200, V(W,B)=10, V(X,C)=10, V(Y,D)=120, V(Z,E)=80, estimar el tamaño de:
a) $\sigma_{A=35 \vee B=5}(W)$
b)$W \ |x| \ X \ |x| \ Y \ |x| \ Z$.

E4. Sea el siguiente log UNDO con checkpoint activo:

```
1. <T1, START>
2. <T1, B, 40>
3. <T2, START>
4. <T2, A, 56>
5. <T2, C, 34>
6. <T3, START>
7. <T1, COMMIT>
8. <T3, B, 12>
9. <T2, COMMIT>
10. <T3, D, 89>
11. <T4, START>
12. <T4, C, 7>
13. <T3, A, 22>
14. <T4, COMMIT>
15. <T3, A, 99>
16. <T3, COMMIT>
```

a) ¿Cómo se guarda un checkpoint que se decide guardar en el momento 5? ¿Cómo sería el registro de start? ¿Dónde? ¿Cómo sería el registro de end? ¿Dónde?
b) El sistema falla después del momento 14. El end checkpoint ya ha sido grabado. ¿Qué parte del log se debe examinar? ¿Qué registros se rehacen en secuencia?
c) Suponer que el log ahora es REDO. ¿Cuál es el momento más temprano para las transacciones T3 y T4 en que los datos son flusheados a disco?

4.1. Resolución

1) Resuelto, Ver ejercicio 1 del 18/12/2013.

2)

```
SELECT C.A
FROM R B, R C, R D
WHERE C.A < D.A and C.A > B.A
GROUP BY C.A
HAVING count(B.A) = count(D.A)
```

3) a) Hago con AND y OR para que sirva para un caso que viene en otro final, y para tirar facha (?):
Si es con un AND:

$$Tam = nW\left(\frac{1}{V(a,W)} \times \frac{1}{V(b,W)}\right) = 100\left(\frac{1}{80} \times \frac{1}{10}\right) = 0{,}125$$

Si es con un OR:

$$Tam = nW\frac{1}{V(a,W)} + nW\frac{1}{V(b,W)} - nW\left(\frac{1}{V(a,W)} \times \frac{1}{V(b,W)}\right)$$

$$Tam = \frac{100}{80} + \frac{100}{10} - 100\left(\frac{1}{80} \times \frac{1}{10}\right) = 11{,}25 - 0{,}125 = 11{,}125$$

Aclaración: En el caso de la AND lo que hago es considerar los eventos como independientes, y por lo tanto puedo multiplicar la probabilidad de ambos sucesos. Luego multiplico por la cantidad de elementos que hay (sería como calcular la esperanza). En el caso de la OR, sumo la cantidad de veces que ocurren ambos sucesos, pero tengo que tener en cuenta que si sólo sumo directo, voy a estar contando dos veces aquellos elementos que cumplan con ambas condiciones, por lo que tengo que restar una vez la cantidad de elementos que cumplen con ambas condiciones (el caso de la AND). Con eso se resuelve ese detalle.

b)

$$Tam = \frac{nW \times nX \times nY \times nZ}{\text{máx}\{V(W,B), V(X,B)\} \times \text{máx}\{V(X,C), V(Y,C)\} \times \text{máx}\{V(Y,D), V(Z,D)\}}$$

$$Tam = \frac{100 \times 400 \times 200 \times 300}{\text{máx}\{10, 200\} \times \text{máx}\{10, 200\} \times \text{máx}\{120, 200\}} = \frac{100 \times 200 \times 300 \times 400}{200 * 200 * 200} = 300$$

Tamaño = 300.

4) a) Suponiendo que se refiere a ponerlo entre el registro 5 y el registro 6 se desplazaría y se agregaría que las transacciones activas son la T1 y T2 (<START CKPT (T1,T2)>). El End se debe poner recién luego del commit de ambas transacciones, recién luego del registro nro 9.

b) Se debe examinar hasta T3 Start, puesto que T1 y T2 ya podemos estar seguros que fueron grabado correctamente. T4 no es necesario de deshacer puesto que se encuentra su commit.

c) Justo después de cada uno de sus commits, que sería justo después de escribir el registro 14 en el log (para T4) y el registro 16 al log (para T3).

5. Final 06/02/2013

E1. Sea R(A,B,C) con dependencias funcionales $F = \{A \to B, B \to C\}$.
a) ¿Está R en FNBC? Si la respuesta es NO, agregar una df a F para que ahora R se encuentre en FNBC.
b) ¿Es posible agregar una df a F para que R quede en 3FN pero no en FNBC?.
Nota: No se considera válido si no hay una correcta explicación de cada paso.

E2. Sean los siguientes esquemas relacionales:
InfoParte(NroParte, NombreParte, Subparte_de)
UsadoEn(NroParte, TipoAvion, CantUsada)
EnStock(NroParte, Lugar, Cantidad)
Responder la siguiente consulta en SQL: *'Para cuáles partes el total en stock es al menos la cantidad usada por un B737 y por qué monto el total excede la cantidad usada'.*
Se deberá tener en cuenta tanto la correctitud como la simplicidad.

E3. Dadas las siguientes cuatro relaciones y sus estadísticas: W, X, Y y Z: Estimar los

W	X	Y	Z
nW = 100	nX = 400	nY = 200	nZ = 300
V(a,W) = 80	V(b, X) = 200	V(c, Y) = 200	V(d,Z) = 200
V(b,W) = 10	V(c, X) = 1	V(d, Y) = 120	V(e,Z) = 80

tamaños de las siguientes expresiones:
a) $\sigma_{a=35 \wedge b=5}(W)$
b) $\sigma_{a=35 \vee b=5}(W)$

E4. Considere la siguiente transacción que se ejecuta en aislamiento:

$$T = L(A); L(B); G(A); G(B); L(C); G(C); G(D)$$

Se produce una caída del sistema y cuando se recupera examinamos el log. Se planean una serie de escenarios.
Para cada escenario, determinar cuales grabaciones de elementos de datos **deben** ser reflejados en la bd en el disco y cuales **NO deben** ser reflejados en el disco. Indicar que un elemento X está en una de esas categorías escribiendo X en el lugar apropiado. Si no hay ninguno en una categoría escribir VACIO.

a) UNDO logging: log = $< TSTART >< T, A, 5 >< T, B, 7 >$
b) UNDO logging: log = $< TSTART >< T, A, 5 >< T, B, 7 >< T, C, 2 >< T, D, 9 >< TCOMMIT >$
c) REDO logging: log = $< TSTART >< T, A, 1 >< T, B, 2 >< T, C, 5 >$
d) REDO logging: log = $< TSTART >< T, A, 1 >< T, B, 2 >< T, C, 5 >< T, D, 0 >< TCOMMIT >$
e) UNDO/REDO logging: log = $< TSTART >< T, A, 5, 1 >< STARTCKPT(T) >< T, B, 7, 2 >< ENDCKPT >< T, C, 2, 5 >< T, D, 9, 0 >< TCOMMIT >$

5.1. Resolución

1) a) La respuesta es NO. $B \to C$ es una df que viola FNBC (B no es clave, A lo es). Hay dos posibilidades para agregar:

1. $C \to A$. De esta manera, todos los atributos son clave, y estan en FNBC.

2. $B \to A$. De esta manera, tanto A como B se determinan mutuamente, y son claves.

b) No, no es posible. Dado que A ya es clave, la primer df no viola FNBC, por lo que queremos que la segunda lo haga (cosa que sucede), pero por el momento también viola 3FN. Para que no viole 3FN y sí FNBC, B no debe ser clave, pero C pertecer a una clave candidata. Es claro que si pusieramos una df que haga que C sea clave, entonces B también lo seria, y no lograríamos lo que esperamos. Poner una df que haga que BC sea clave tampoco serviría, porque como $B \to C$, entonces BC es superclave, siendo B clave mínima, y volvemos a estar en FNBC. No podemos poner ninguna df tal que AC sea clave, pues A ya lo es. El mismo razonamiento vale para ABC.

2)

```
SELECT u.NroParte, s.Cantidad - u.CantUsada as Diferencia
FROM UsadoEn u, EnStock s
WHERE u.NroParte = s.NroParte and s.Cantidad >= u.CantUsada and u.TipoAvion =
    'B737';
```

Podría agregarse un 'DISTINCT' en el select de considerarse necesario. Además, si por alguna razón se pidiera cuál es la máxima diferencia con algún Boeing (no es lo que pide) habría que agrupar por Nro de parte y pedir Max(s.Cantidad - u.CantUsada), y nada más (el 'where' quedaría de la misma forma).

3) Idem Ej3 del 24/07/2013.

4) No entendí lo de la 'X', pero sacando eso:
a) Debe quedar: A = 5, B = 7. No Debe: Vacio.
b) Debe quedar: VACIO. No Debe quedar: A = 5, B = 7, C = 2, D = 9
c) Debe quedar: VACIO. No debe quedar: A = 1, B = 2, C = 5
d) Debe quedar: A = 1, B = 2, C = 5, D = 0. No debe quedar: VACIO
e) Debe quedar: A = 1, B = 2, C = 5, D = 0. No debe quedar: A = 5, B = 7, C = 2, D = 9

6. Final 15/08/2012

E1. Dada la relación $R = (A, B, C, D, E, G, H)$ y el conjunto de dependencias $F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$.
Cuál de las siguientes descomposiciones de R:

I. $\{AB, BC, ABDE, EG\}$

II. $\{ABC, ACDE, ADG\}$

a) Preserva las dependencias.

b) Es una descomposición sin pérdida de información.

E2. Sean los siguientes esquemas relacionales:
Vuelo(VueloNro, Desde, Hacia, Distancia, Partida, Arribo, Precio)
Aeronave(AId, ANombre, Rango)
Certificado(EId, AId)
Empleado(EId, Enombre, Sueldo)

La relación **Empleado** contiene datos de todos los empleados de la compañía, entre ellos los pilotos.
En la relación **Certificado** solo figuran los pilotos certificados para volar una determinada aeronave.
Responder la siguiente consulta en SQL:
'Listar los nombres de los pilotos que pueden volar aeronaves con rango de crucero mayor a 5000 millas pero que no están certificados en ningún avión Boeing'

3) Describir detalladamente el método de junta con índice de agrupamiento y dar su costo.

4) La siguiente es la matriz de compatibilidad para tres modos de locks hipotéticos X, Y y Z.
Recordar que se puede otorgar un lock sobre un elemento A de la base de datos en el modo representado por columna j si y solo si no hay ninguna otra transacción que tiene un lock sobre A en algún modo i, donde la posición i,j de la matriz tiene un "no".

	X	Y	Z
X	Yes	Yes	No
Y	No	Yes	Yes
Z	Yes	No	Yes

De las tres afirmaciones siguientes:
1. Es posible para diferentes transacciones mantener locks sobre el mismo elemento de base de datos en los modos X y Z al mismo tiempo
2. Es posible para más de una transacción mantener un lock sobre algún elemento de la base de datos en modo Z, al mismo tiempo que otra transacción mantiene un lock sobre el mismo elemento en modo Y.

3. Es posible para diferentes transacciones mantener locks sobre el mismo elemento en los tres modos de lock al mismo tiempo.

¿Cuáles son verdaderas?
a) 1 solamente
b) 2 solamente
c) 1 y 2 solamente
d) 1 y 3 solamente
e) 1, 2 y 3

6.1. Resolución

1) Suponiendo que la H no existe (sino, no tiene sentido el ejercicio, y obviamente hay pérdida de información por todas partes), aplicamos los algoritmos de Chase y el que permite analizar si no hay péridas de dependencias funcionales:
a) No preserva las dependencias funcionales (Se pierde $AB \rightarrow C$). Es una descomposición con perdida de información.
b) No preserva las dependencias funcionales (Se pierde $B \rightarrow D$). Es una descomposición sin perdida de información.

2)

```
SELECT Enombre
FROM Empleado
WHERE EId in (
   SELECT Eid
   FROM Certificado c, Aeronave a
   WHERE  c.Aid = a.Aid and a.rango > 5000
   ) and EId not in(
   SELECT Eid From Certificado c, Aeronave a
   WHERE c.Aid = a.Aid and a.nombre <> "Boeing"
   );
```

3) Join con Indice de agrupamiento, ver ej3 del final del 18/12/2013.

4) Luego de leer detenidamente la tabla, podemos ver que dependiendo del orden, podemos llegar a poner una serie de locks sobre un mismo elemento. Analizamos cada una de las afirmaciones:

1. La primer afirmación es correcta, puesto que puedo poner sobre un mismo elemento primero un lock X, y luego un Lock Z (en el orden inverso esto no se podria).

2. La segunda afirmación también es cierta, puesto que podemos poner sobre un elemento un lock Z y luego un lock Y (en el orden inverso no se podria, pero no es estricto con ésto).

3. La tercera afirmación es falsa, puesto que, aunque puedo encontrar varias secuencias de pares de locks válidos, es imposible encontrar una tal que el tercer lock a poner no esté en

conflicto con los dos anteriores (tener en cuenta que los locks se van a aplicar sobre el mismo elemento al mismo tiempo).

Por lo tanto, la respuesta correcta es la **C**. Mucho no tenía que ver con la materia, pero bueno.

7. Final 08/08/2012

E1. Sea $R = (A, B, C, D, E)$ y $F = \{A \to B, B \to C, C \to AD, D \to E, E \to A\}$. Normalizar a Boyce-Codd el esquema que resulta de proyectar R sobre ACE.

E2. Estimar el tamaño de la junta $R(A, B)|x|S(B, C)$ si se cuenta con el siguiente histograma:

	$B < 0$	$B = 0$	$B > 0$
R	50	10	40
S	30	20	50

Asumir que existen 10 valores diferentes de $B < 0$ y 20 valores diferentes de $B > 0$.

E3. Dado el siguiente esquema relacional:
Estudiante(enum,enombre,carrera,año,edad)
Clase(cnombre,horario,aula,Pid)
Cursa(enum,cnombre)
Profesor(Pid,pnombre,deptid)

Escribir una consulta SQL que permita, para cada valor de edad que aparece en Estudiante, hallar el valor de año que aparece más frecuentemente. Por ejemplo, si hay más estudiantes de 20 años de edad en 2do año que en cualquier otro año para estudiantes de 20 a, listar el par (20,2do).
Nota: no deben listarse duplicados y la consulta debe ser tan concisa como resulte posible.

E4. Sea el siguiente log:

```
<T1, START>
<T1, X, 5, 10>
<T2, START>
<T2, X, 10, 20>
<T1, COMMIT>
<T2, Y, 30, 60>
<START\ CKPT (T2)>
<T2, W, 35, 70>
<T3, START>
<T3, W, 70, 40>
<END\ CKPT>
<T2, Z, 20, 40>
<T2, COMMIT>
<T3, Z, 40, 80>
```

El mecanismo de control de concurrencia es Locking de dos fases y solo hay locks de lectura (SL) y de grabación (XL).
a) Dado los supuestos, es posible que el log tenga los registros indicados? Explicar. Si la respuesta es NO, cuál es el primer registro "imposible" de log?, por qué? Eliminar ese registro. Es

posible que el log contenga la nueva secuencia? Nuevamente explicar por qué SI o por qué NO. Repetir hasta obtener una secuencia posible de registros de log.
b) Para la secuencia obtenida en a), cuáles son los posibles valores de X, Y, W y Z después que el último de esos registros es grabado al disco y antes de la recuperación? Explicar.

7.1. Resolución

1) Proyectar $R' = \pi_{A,C,E}(R) = (A, C, E)$, $F' = \{A \to C, C \to E, E \to A\}$ (Se deducen de las otras). Ya se encuentra en FNBC. Fin del ejercicio :P

2)

$$Tamanio = Tam < 0 + Tam = 0 + Tam > 0$$

$$Tam = 0 = 10 \times 20 = 200$$

$$Tam < 0 = \frac{50 \times 30}{\text{máx}\{10, 10\}} = 150$$

$$Tam > 0 = \frac{40 \times 50}{\text{máx}\{20, 20\}} = 100$$

$$\Rightarrow Tamanio = 450$$

3)

```
SELECT DISTINCT e.edad, e.anio
FROM Estudiante e
GROUP BY e.edad, e.anio
HAVING count(*) >= ALL (SELECT COUNT(*)
        FROM Estudiante e2
        WHERE e2.edad = e.edad
        GROUP BY e2.anio);
```

4) a) La primera operacion de T2 es imposible (porque T1 esta modificando X, y T2 tambien quiere hacerlo). Eliminando esa entrada, aun es imposible porque T2 modifica W, y T3 tambien desea hacerlo en simultaneo. Eliminando ese segundo registro, el log pasa a ser posible.

b) Para antes de la recuperacion: Solamente se sabe que todas las operaciones anteriores al start-ckpt fueron escritas a disco. Por lo tanto, sólo podemos asegurar que X = 10, Y = 60, W y Z no podemos estar seguro de su contenido antes de la recuperación (justamente por ésto es tan importante la etapa de recuperación).

Despues de la recuperación:
Al recuperar, se deshacen las operaciones no finalizadas (yendo desde el último registro del log, hasta el primero, o bien yendo hasta el último Start para cada Registro que se encontrara en un END Checkpoint), luego se rehacen las finalizadas.

- X = 10 (Todas las operaciones sobre X son finalizadas, y este es su valor dado que el otro registro se elimino)
- Y = 60 (Todas las operaciones sobre Y son finalizadas)

- W = 70 (Todas las operaciones sobre W son finalizadas, recordando que se elimino la modificacion realizada por T3)
- Z = 40 (Se deshace el cambio realizado por T3 dado que no encontramos el commit).

Ademas se escribe al log la entrada $< ABORT\ T3 >$ (con el flush correspondiente).

8. Final 15/02/2012

E1. Sean $R(A, B, C, D)$ y $F = \{A \to B, B \to C, D \to B\}$ Se quiere descomponer a R para que se encuentre en FNBC.
a) Se elige descomponer en ACD y BD. Dar un cubrimiento minimal para ambas tablas.
b) ¿Está ACD en FNBC? Si no lo está, dar una descomposición para ACD que lo esté.

E2. Sea $R(A)$ escribir una consulta SQL que calcule la mediana de un conjunto de números, sabiendo que no hay valores repetidos, no hay valores NULL y se puede asumir que la cantidad de tuplas es impar. (Se considera tanto la correctitud como la simplicidad de la consulta).

E3. Sean 4 tablas: AB, BC, CD, DE. Se tienen todos los nAB, nBC, ..., nDE y todos los V(B, AB) ... V(D, DE).
a) Calcular la cantidad total de árboles de consulta que existen. Elegir entre 0, 1, 2, 6 y 8.
b) Determinar cuál es el árbol de menor costos de los calculados en el punto a).

E4. Dado el siguiente log:

```
(T1, START)
(T2, START)
(T3, START)
(T1, A, 0, 1)
(T2, A, 1, 2)
(T3, A, 2, 3)
(T2, COMMIT)
```

Luego de la recuperación, el valor de A en disco (no hay cascada de rollbacks) es: 0, 1, 2, 3 o No se puede determinar en base a la información disponible.

8.1. Resolución

1) Siendo $R1 = (A, C, D)$, $R2 = (B, D)$:
a) $F1 = \{A \to C, D \to C\}$; $F2 = \{D \to B\}$.
b) ACD no esta en FNBC, propongo: $R3 = (A, C)$, $F3 = \{A \to C\}$, $R4 = (AD)$, $F4 = \emptyset$. (Se pierde la dependencia $D \to C$).

2) Igual al ej2 del 24/07/2013.

3) Los árboles de consultas son los árboles que genera el planificador de la consulta para poder realizarla de la manera más óptima. Por lo tanto, es obvio que intentará ir realizando las juntas a medida que avanza (no intentará hacer producto carteciano entre AB y CD como primera instancia). Además, para poder realizar las optimizaciones de la manera más sencilla, esos árboles tienen una particularidad: son sesgados, por lo tanto, el hijo derecho es necesariamente una hoja. Por lo tanto, sólo puede hacer joins de tipo 'pipeline'. Además, por lo que tengo entendido, no es lo mismo hacer $AB|x|BC$ que $BC|x|AB$, simplemente porque no es *el mismo* árbol (son dos distintos, que es lo que estamos contabilizando), por lo tanto, tenemos las siguientes opciones:

1. $((AB|x|BC)|x|CD)|x|DE$
2. $((BC|x|AB)|x|CD)|x|DE$
3. $((BC|x|CD)|x|AB)|x|DE$
4. $((BC|x|CD)|x|DE)|x|AB$
5. $((CD|x|BC)|x|AB)|x|DE$
6. $((CD|x|BC)|x|DE)|x|AB$
7. $((CD|x|DE)|x|BC)|x|AB$
8. $((DE|x|CD)|x|BC)|x|AB$

Como podemos ver, son 8 árboles (ver que si $AB|x|BC$ fuera el mismo árbol que $BC|x|AB$, serían nada más 4, que no es un opción posible).
Sobre cuestión de costos, no puedo estar totalmente seguro, pero el sentido común me indica que lo mejor es tener de izquierda a derecha las relaciones de tal manera que las juntas vayan dando la menor cantidad de elementos, por lo tanto, teniendo todos los datos a mano, se pueden ir haciendo las cuentas: Agarramos cada posibilidad, y vamos calculando de izquierda a derecha el número de tuplas que van teniendo (vamos calculando la estimación junta a junta). Obviamente el resultado final debería dar lo mismo, lo que nos interesan son los resultados intermedios. Nos vamos quedando con aquella solución que va teniendo mejores resultados. Por ejemplo, para calcular el del primer caso, paso a paso:

1. $n(AB|x|BC) = nABC = \frac{nAB}{V(B,AB)} \times \frac{nBC}{V(B,BC)} \times \text{mín}\{V(B,AB), V(B,BC)\}$.
2. $n((AB|x|BC)|x|CD) = nABCD = \frac{nABC}{V(C,BC)} \times \frac{nCD}{V(C,CD)} \times \text{mín}\{V(C,BC), V(C,CD)\}$
3. $n(((AB|x|BC)|x|CD)|x|DE) = nABCDE = \frac{nABCD}{V(D,CD)} \times \frac{nDE}{V(D,DE)} \times \text{mín}\{V(D,CD), V(D,DE)\}$

Se puede ir haciendo algún tipo de ranking, quedándo con la mejor opción. También hay que tener en cuenta los costos de cada método de junta que se vaya haciendo (por eso no es lo mismo hacer cualquiera de las dos opciones de una conmutativa, porque como se vio en clase, no todos los métodos de juntas tienen costos 'simétricos' o conmutativos, entonces también se puede ponderar por eso).

4) El log es claramente un UNDO/REDO. Dado Que la transaccion 2 tiene su commit, pero ninguna de las otras 2 tiene sus respectivos commits, el valor final será: 2.

9. Final 21/12/2011

1) Dados$R(A, B, C, D)$ y la dependencia de junta $J = |x|[AB, BC, CD]$
a) Dar una instancia de R que muestre que $M = C \rightarrow\rightarrow A$ no puede inferirse de J.
b) Usando tableau y el algoritmo chase mostrar que M puede inferirse a partir de J y de $N = D \rightarrow\rightarrow B$.

2) Dado $R(A)$ siendo A números que pueden estar repetidos y ninguno es NULL escribir una consulta SQL que devuelva la moda de A (la moda es el valor más frecuente).

3) Calcular el tamaño de la junta AB $|x|$ BC $|x|$ CD $|x|$ DE.
Datos:
nAB = 100, nBC = 200, nCD = 300, nDE = 400
V(A,AB)=20, V(B,AB)=50, V(B,BC)=50, V(C,BC)=40, V(C,CD)=60, V(D,CD)=100, V(D,DE)=50, V(E,DE)=100.

4) Dados los atributos A y B, ambos con valor 0 al principio, una transacción modifica ambos y les pone el valor 1. Se genera el siguiente log:

```
<START T>, <T, A, 0>, <T, ?, ?>, <COMMIT T>
```

Decir qué tipo de log es, y completar los "?".

9.1. Resolución

1) a) Una posible instancia sería:

```
a1 b1 c1 d1
a1 b1 c1 d2
a2 b2 c1 d1
a2 b2 c1 d2
```

Se ve que cumple con las condiciones ya que la junta natural de la proyección sobre cada uno da la relación original:

```
a1 b1    |x|  b1  c1   = a1 b1 c1 |x|  c1  d1 =
a2 b2         b2  c1   = a2 b2 c1      c1  d2

a1 b1 c1 d1
a1 b1 c1 d2
a2 b2 c1 d1
a2 b2 c1 d2
```

¿Se cumple $C \rightarrow\rightarrow A$? ver que si agarro las tuplas 1 y 3, debe existir una tupla con c = c1, a = a1, b = b2 y d = d1, la cual no existe, por lo tanto no se puede deducir M = $C \rightarrow\rightarrow A$ a partir de la dj mencionada.

b) Queremos ver que se cumple $C \twoheadrightarrow A$ si tenemos ademas $N = D \twoheadrightarrow B$. Para esto planteamos la descomposición $R1 = AC$ y $R2 = BCD$.

	A	B	C	D
AC	a_1	b_{12}	a_3	b_{14}
BCD	b_{21}	a_2	a_3	a_4

Aplicamos la dependencia de junta $|x|[AB, BC, CD]$, proyectando sobre las columnas y realizando la junta:

A	B
a_1	b_{12}
b_{21}	a_2

B	C
b_{12}	a_3
a_2	a_3

C	D
a_3	b_{14}
a_3	a_4

Haciendo la junta natural entre estas tablas:

A	B	C	D
a_1	b_{12}	a_3	b_{14}
b_{21}	a_2	a_3	b_{14}
a_1	b_{12}	a_3	a_4
b_{21}	a_2	a_3	a_4

Las tuplas 1 y 4 ya se encontraban en la tabla inicial, entonces no las volvemos a poner:

	A	B	C	D
AC	a_1	b_{12}	a_3	b_{14}
BCD	b_{21}	a_2	a_3	a_4
	b_{21}	a_2	a_3	b_{14}
	a_1	b_{12}	a_3	a_4

Como vemos, con solo esto no nos queda una fila con todos elementos distinguidos, lo cual corrobora que $C \twoheadrightarrow A$ no se puede deducir de la dependencia de junta únicamente. Aplicamos la dmv N, transformándola en una dependencia de junta $|x|[BD, ACD]$.

B	D
b_{12}	b_{14}
a_2	a_4
a_2	b_{14}
b_{12}	a_4

A	C	D
a_1	a_3	b_{14}
b_{12}	a_3	a_4
b_{21}	a_3	b_{14}
a_1	a_3	a_4

Haciendo la junta natural obtenemos:

A	B	C	D
a_1	b_{12}	a_3	b_{14}
b_{21}	b_{12}	a_3	b_{14}
b_{21}	a_2	a_3	a_4
a_1	a_2	a_3	a_4
a_1	a_2	a_3	b_{14}
b_{21}	a_2	a_3	b_{14}
b_{21}	b_{12}	a_3	a_4
a_1	b_{12}	a_3	a_4

Como conseguimos toda una fila con elementos distinguidos (fila 4), podemos ver que M se puede deducir a partir de J y N.

2) Consulta de moda:

```
SELECT A
FROM R
GROUP BY A
HAVING Count(*) >= ALL ( SELECT Count(*) FROM R GROUP BY A);
```

3) El tamaño de la junta estara dado por:

$$Tam = \frac{nAB \times nBC \times nCD \times nDE}{\text{máx}\{V(B,AB), V(B,BC)\} \times \text{máx}\{V(C,BC), V(C,CD)\} \times \text{máx}\{V(D,CD), V(D,DE)\}}$$

$$Tam = \frac{100 \times 200 \times 300 \times 400}{\text{máx}\{50,50\} \times \text{máx}\{40,60\} \times \text{máx}\{100,50\}} = \frac{100 \times 200 \times 300 \times 400}{50 \times 60 \times 100} = 8000$$

$$\Rightarrow Tam = 8000$$

4) El log únicamente podría ser de tipo UNDO. Si la consigna no especificara el valor que se le quiere asignar a los datos, podría llegar a darse el caso que pueda ser de tipo REDO (y se le esté asignando el valor 0 a A, por lo tanto, el mismo que tenía antes). No siendo éste el caso, podemos decir rápidamente que la otra transacción debe ser de la forma <T, B, 0>, pues la consigna especifica que se quiere modificar ambos A y B, y al ser de tipo UNDO, se guarda el valor anterior en el registro del log (0 en este caso).

10. Final 13/07/2011

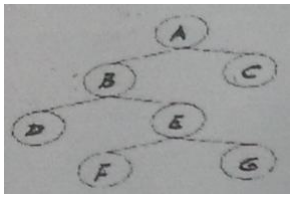
E1. Dada una relación dada por $R(A, B, C, D, E)$ y $M = \{A \twoheadrightarrow BC, D \rightarrow C\}$:
a) ¿Qué otra dependencia funcional se cumple?
b) Dar una instancia de R que satisfaga esas dfs y dmvs.

E2. Sean los siguientes esquemas relacionales:
Estudiante (Enro, Enombre, Carrera, Anio_cursa, Edad)
Curso (Cnombre, Horario, Aula, Pid)
Cursa (Enro, Cnombre)
Profesor(Pid, Pnombre, departamento)

Escribir una consulta en SQL que resuelva: *'Hallar los nombres de todos los profesores que enseñan en todas las aulas en las que se dicta algún curso'.*

E3. Describir el método MERGE (a nivel pseudocódigo) para realizar la junta de $R \mid x \mid S$ y exprese su costo.

E4. Dado el siguiente árbol de elementos lockeables:



Considere las siguientes transacciones que operan sobre el árbol:

```
T1;L1(A);L1(B);L1(E)
T2;L2(A);L2(C);L2(B)
T3;L3(B);L3(E);L3(F)
```

Si los planes de ejecución siguen el protocolo de árbol, de cuántas formas diferentes pueden realizarse:
a) T1 y T2
b) T2 y T3

10.1. Resolución

1) $R(A, B, C, D, E) M = \{A \twoheadrightarrow BC, D \rightarrow C\}$
a) Por el axioma de interacción, como tenemos $A \twoheadrightarrow BC$, y además tenemos que $D \cap BC = \emptyset$, tenemos a $C \subset BC$ y $D \rightarrow C$, $\Rightarrow A \rightarrow C$.

b)

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_2	c_1	d_2	e_2
a_1	b_1	c_1	d_2	e_2
a_1	b_2	c_1	d_1	e_1

2)

```
SELECT P.Pnombre
FROM Profesor P
WHERE not exists (SELECT *
       FROM Curso
       WHERE C.Aula not in (SELECT c2.Aula
                 FROM Curso c2
                 where p.pid = c2.pid));
```

Lease como: obtener los nombres de profesores para los cuales no exista un aula en las que se dan cursos que no esté entre las que utiliza el profesor para dar un curso. También es válida la siguiente, con un doble 'not exists':

```
SELECT p.pNombre
FROM Profesor p
WHERE NOT EXISTS
(
   (SELECT Aula
   FROM Curso c1
   WHERE NOT EXISTS
      (SELECT *
      FROM Curso c2
      WHERE c1.aula = c2.aula AND c2.pID = p.pID)
      )
);
```

3) El método Merge consiste en tener ordenadas ambas tablas (por los atributos del join), y luego aprovechar ese ordenamiento para leer de forma secuencial las tuplas e ir realizando el join. En el caso que las tablas ya se encontraran ordenadas, el costo sería únicamente de la cantidad de bloques de cada una de las tablas ($Br + Bs$), en caso que fuera necesario ordenarlas, se agrega los costos de realizarlo por el método más conveniente (Sort-Merge).

4) a) T1 y T2 no pueden laburar en paralelo porque lo primero que hacen es lockear A y no lo liberan hasta terminar la secuencia, por lo que no es posible acceder. Por lo tanto solo hay dos opciones (T2 y T1, o T1 y T2).
NOTA: Luego de haber leido bien sobre el tema pude contestar la B, y un día antes del final me dí cuenta que esta respuesta está mal, si hay combinaciones posibles para hacerlas en paralelo (no tantas como en el punto b, pero las hay), pero ya fue :P si alguien quiere corregir y ponerlas que lo haga...

b)

- T2 y luego T3
- T3 y luego T2
- L2(A), T3, L2(C), L2(B)
- L2(A), L2(C), T3, L2(B)
- L2(A), L3 (B), L2(C), L3(E), L3(F), L2(B)
- L3(B), L2(A), L2(C), L3(E), L3(F), L2(B)
- L3(B), L3(E), T2, L3(F)
- L3(B), L2(A), L3(E), L2(C), L2(B), L3(F)
- L3(B), L3(E), L2(A), L3(F), L2(C), L3(F)

Puede que falte alguna, pero no creo.

11. Final 06/06/2011

E1. Dada el esquema de relación $R = (A, B, C, D)$ y el conjunto de dependencias funcionales $F = \{AB \to C, BD \to C, A \to D\}$.
a) Hallar un cubrimiento minimal F_M para F.
b) Hallar una descomposición de R que sea 3FN y preserve las dependencias en tan sólo dos esquemas.
c) Indicar qué dependencias funciones se proyectan.
d) Indicar si se preserva la información.
e) La descomposición de b) ¿está en FNBC?, si no lo está, halle una descomposición de R que si lo esté.

E2. Hallar la cardinalidad (o cantidad de tuplas del resultado) de $(((R\ |x|\ S)\ |x|\ T)\ |x|W)$ sabiendo que: (Faltarían los datos sobre C, que no aparecen por ningún lado).

R(A,B,C)	S(A,B,D)	T(A,C,D)	W(B,C,D)
nR = 1000	nS = 2000	nT = 3000	nW = 4000
V(A,R) = 1000	V(A,S)=50	V(A,T)=50	-
V(B,R)=50	V(B,S)=100	-	V(B,W)=40
-	V(D,S)=200	V(D,T)=500	V(D,W)=400

E3. Dados los siguientes esquemas de relación: **Estudiante (Enro, Enombre, Carrera, Anio_cursa, Edad)**
Curso (Cnombre, Horario, Aula, Pid)
Cursa (Enro, Cnombre)
Profesor(Pid, Pnombre, departamento)
Crear una consulta que obtenga, para una dada edad, el año que más frecuente. Es decir, por ejemplo, para el caso de 20 años, si la mayoría de los estudiantes con esa edad cursan el 2do año desu carrera, entonces la consulta debe devolver (20, 2do).
Nota: La consulta debe ser lo más acotada posible.

E4. Dada la siguiente ejecución:

```
L1(A), L2(B), L3(C), L1(B), L2(C), L3(D), G1(A), G2(B), G3(C).
```

a) Introducir los locks de lectura simultánea, locks exclusivos, locks de update y la liberación de los mismos.
b) Describir detalladamente que sucede cuando el scheduler realiza la ejecución de E obtenida en a)

11.1. Resolución

1) a) **Paso 1**: los lados derechos tienen un solo atributo. Este paso ya está cumplido, así que no hacemos nada.
Paso 2: Vemos que los lados izquierdos no sean redundantes. Para esto calculamos las clausuras de sub elementos del lado izquierdo, para ver si se puede llegar a quitar algun atributo. $A \to D$ está perfecta, así que no la tocamos.

- $AB \to C$. Vemos $(A)^+ = \{A, D\}$, entonces no es redundante B. Vemos $(B)^+ = \{B\}$, entonces A no es redundante.
- $BD \to C$. Vemos $(B)^+ = \{B\}$ entonces D no es redundante. Vemos $(D)^+ = \{D\}$ entonces B no es redundante.

Por lo tanto, en el paso 2 no fue necesario corregir nada.
Paso 3: vemos que ninguna df sea redundante (i.e. se pueda conseguir de, por ejemplo, una transitividad). Para esto eliminamos cada df y vemos si la clausura del lado izquierdo sigue siendo la misma:

- Saco $AB \to C$. $(AB)^+ = \{A, B, D, C\}$. Vemos que la clausura sigue conteniendo a C, por lo tanto la df era redundante.
- Saco $BD \to C$. $(BD)^+ = \{B, D\}$, por lo tanto la df no es redundante.
- Saco $A \to D$. $(A)^+ = \{A\}$, por lo tanto la df no es redundante.

Finalmente: $F_{min} = \{BD \to C, A \to D\}$.

b) Justamente 3FN va a mantener las df, siempre que apliquemos bien la descomposición, y como sólo tenemos dos df en F_{min} (i.e. 2 no triviales):
$R_1 = (A, D); F_1 = \{A \to D\}$
$R_2 = (B, C, D) F_2 = \{BD \to C\}$
$R_3 = (AB) F_3 = \emptyset$ (Esta tiene que estar porque sino no está la clave en ninguna!).

c) Ya se respondió.

d) Por haber utilizado el algoritmo de descomposición de 3FN, sí o sí se debe preservar la información.

e) La descomposición realizada se encuentra además en FNBC.

2) Ejercicio resuelto varias veces, ver Ej3 del 24/07/2013 (puede que sea con otros valores).

3) Ejercicio de la moda de las edades de estudiantes. Resuelto en Ej3 del 08/08/2012.

4) a)

```
E = L1(A), L2(B), L3(C), L1(B), L2(C), L3(D), G1(A), G2(B), G3(C)
```

Luego de agregar los locks y unlocks, recordando que tenemos que preservar el protocolo de locking de 2 fases (Primero se piden los locks, y luego se hacen los unlocks, no se puede lockear algo despues de haber unlockeado cualquier elemento):

```
E = XL(A, T1),SL(B, T1), L1(A), UL(B, T2),SL(C,T2), L2(B),
UL(C, T3)SL(D, T3), L3(C),L1(B) ,L2(C), L3(D), G1(A),
UNLOCK(A, T1), UNLOCK(B, T1), G2(B), UNLOCK (B, T2), UNLOCK (C, T2),
G3(C), UNLOCK(C, T3), UNLOCK (D, T3)
```

b) La primera transacción es T1, que pide sus locks, necesita uno para poder escribir sobre A por lo que se agrega un lock de escritura o exclusive lock (un lock de update se comportaría de la misma manera, siendo que no hay otras transacciones operando sobre A), y también se agrega un lock de lectura (o share lock) para B, pues lo único que hará la transacción es leer tal valor. Luego se prosigue a realizar la primera operación de T1, que es la lectura de A. Luego, como está por aparecer una nueva transacción, se piden los locks correspondientes: como va a modificar a B, es necesario agregar un lock de update para éste, por lo cual deberá esperar a que T1 haga el Unlock correspondiente para poder grabar a B. No se podría poner un exclusive lock porque hay un SL sobre B, y no correspondería con la ejecución del enunciado. Además, como se leerá C se agrega un lock de lectura sobre ese dato. Luego se lee el valor de B. Se realiza lo mismo para la transacción T3, utilizando el mismo criterio ponemos un update lock para C y un share lock para D. Luego se prosigue realizando las operaciones (lectura de C en T3, lectura de B en T1, lectura de C en T2, lectura de de D en T3), llegamos a la operación G1(A), y luego de realizar tal grabación, pasamos a realizar los UNLOCKS, por lo tanto, ahora T2 si puede realizar la grabación de B (Su update lock puede verse ahora como un XL). Luego de realizar la grabación de B, se realizan los Unlocks, por lo que ahora T3 puede realizar el grabado de C. Luego de ésto, se realizan los últimos unlocks terminando con la ejecución.

12. Final 16/02/2011

E1. Dar un ejemplo de una relación $R(A, B, C, D, E)$ que satisface la dj $|x|[ABC, BDE, ACE]$ pero no satisface ninguna dmv trivial.

E2. Dadas las relaciones:
Estudiante (Enro, Enombre, Carrera, Anio_cursa, Edad)
Curso (Cnombre, Horario, Aula, Pid)
Cursa (Enro, Cnombre)
Profesor(Pid, Pnombre, departamento)
Escribir en SQL la consulta: *'Hallar los nombres de los estudiantes que cursan el máximo numero de cursos'*

E3. Describir el método PIPELINE para realizar la junta múltiple $R1$ $|x|$ $R2$ $|x|$ $R3$ $|x|$ $R4$

E4. Se obtuvo el siguiente LOG. Se sabe que es UNDO/REDO y que utiliza locking de dos fases: de lectura (SL) y de grabado (XL):

```
<T1, START>
<T1, X, 5, 10>
<T2, START>
<T2, X, 10, 20>
<T1, COMMIT>
<T2, Y, 30, 60>
<START CKPT (T2)>
<T2, W, 35, 70>
<T3, START>
<T3, W, 70, 40>
<END CKPT>
<T2, Z, 20, 40>
<T2 COMMIT>
<T3, Z, 40, 80>
```

a) Es posible que el log tenga el contenido descripto arriba, dados los supuestos indicados del sistema? Si NO, ¿Cuál es el primer registro "imposible" del log? ¿Por qué? ELIMINAR ESE REGISTRO. ¿Es posible que ahora el log contenga una nueva secuencia válida? Explicar por qué SI o NO. Repetir hasta lograr una secuencia posible de registros de log.
b) Dada la secuencia resultante en (a), ¿cuáles son los valores posibles de X, Y, W, Z después que el último registro del log es guardado a disco y ANTES de la recuperación? Explicar por qué.

12.1. Resolución

1) Tenemos $R(A, B, C, D, E)$ y $J = |x|(ABC, BDE, ACE)$.
Una instancia de R que cumpla con tal restricción debe cumplir que:

```
T1[B] = T2[B]
T2[E] = T3[E]
```

```
T3[AC] = T1[AC]
T4[B] = T1[B]
T4[E] = T2[E]
T4[AC] = T3[AC]
```

Ponemos entonces:

	A	B	C	D	E
t_1	a_1	b_1	c_1	d_1	e_1
t_2	?	b_1	?	?	e_2
t_3	a_1	?	c_1	?	e_2
t_4	a_1	b_1	c_1	?	e_2

En donde hay ? es que puede ir cualquier cosa, que se seguirán cumpliendo las restricciones. Ahora bien, no queremos que se cumpla ninguna dmv trivial. Recordemos que $X \twoheadrightarrow Y$ es trivial si $X \cup Y = R$, o $X \supseteq Y$.
Eso es totalmente imposible, justamente una dmv trivial es trivial porque siempre se cumple. Supongamos que queremos probar que $ABCD \twoheadrightarrow E$ no se cumple. Entonces tenemos que poner dos tuplas con el mismo ABCD, con distinto E y distinto 'resto' r1 y r2, por lo que deberían existir otras dos pares de tuplas que tengan ABCD, cada una con el E correspondiente, y con un par de restos iguales a r1 y r2, pero como r1 y r2 no son nada, tampoco pueden ser distintos entre si, por lo tanto esas tuplas son las originales, por lo que es verdad que teniendo dos tuplas con mismo ABCD y distinto E, se cumple la dmv trivial. Si tuvieramos solo una tupla, con ABCDE, tambien se cumple, ya sea porque la definición contempla una implicación en la cual del lado izquierdo pedimos que los E sean distintos ($Falso \rightarrow Verdadero$ es verdadero), o bien porque quitando ese detalle de definición, la misma tupla sería la segunda y tercer tupla, cumpliendo lo pedido.
Por lo tanto, indistintamente de las restricciones que pongamos, no es posible que no se cumpla esa dmv trivial (ver que en realidad no depende de si se llama ABCD y E, sino que se puede generalizar). Por lo tanto lo pedido es imposible.

2)

```sql
SELECT DISTINCT e.eNombre
FROM Estudiantes e
WHERE e.eNro IN (
   SELECT c.eNro
   FROM Cursa c
   GROUP BY c.eNro
   HAVING COUNT(*) >= ALL (
      SELECT COUNT(*)
      FROM Cursa c2
      GROUP BY c2.eNro
   )
);
```

3) El método pipeline se utiliza para optimizar la junta de más de dos tablas. Teniendo en cuenta que tenemos muchas formas de realizar la junta (sabiendo que las operaciones de junta

son conmutativas y asociativas), lo que se busca evitar es volver a leer dos veces un mismo resultado, por lo que éste método optimiza en base a ir utilizando los resultados intermedios en las siguientes juntas, evitando escribir a disco hasta llegar a los resultados finales. Obviamente, no cualquier combinación de las tuplas será necesariamente beneficiada de utilizar este método. En general se ven beneficiadas las combinaciones de paréntesis del tipo "cascada". El algoritmo utilizado se basa en utilizar ciclos anidados (o en realidad, ciclando de manera recursiva) para el cual se busca (para el caso del ejemplo) por cada tupla de R_1, agarrar las tuplas de R_2 que sirvan para la junta, realizando esta junta parcial. Inmediatamente a continuación, por cada tupla de éste resultado parcial, buscamos las tuplas en R_3 que puedan juntarse con tal tupla, y se realizan las juntas, obteniendo un nuevo resutlado parcial. Igual al caso anterior, inmediatamente por cada tupla del resultado parcial obtenido, buscamos las tuplas de R_4 que pueden hacer la junta con tal tupla, y finalmente estas tuplas se van escribiendo como resultado final. Aclaración: Para realizar cada 'junta' en sí, se puede usar cualquier algoritmo de los vistos en clase.

4) Ejercicio identico al del 08/08/2012

13. Final 14/07/2010

E1. ¿Cuál de los siguientes es un contraejemplo para demostrar que si $A \twoheadrightarrow BC \Rightarrow A \twoheadrightarrow B$ es falso?
Justificar porque los otros casos no sirven como contraejemplo.
a)

A	B	C
1	11	21
1	12	22

b)

A	B	C
1	11	21
1	12	22
1	11	22
1	12	21

c)

A	B	C	D
1	11	21	31
1	12	22	32

d)

A	B	C	D
1	11	21	31
1	12	22	32
1	11	21	32
1	12	21	31

E2. Estimar el tamaño de la junta $R(A, B)|x|S(B, C)$ utilizando histogramas.
Asumir V(B,R) = 20 y V(B,S) = 21

	0	1	2	3	4	Otros
R.B	5	6	4	5	?	48
S.B	8	9	5	?	7	68

Donde dice ? es que se desconoce la cantidad (entra dentro de la categoría de 'Otros').

E3. Dada la consulta sobre R(A, B, C, D), asumiendo que A, B, C y D son enteros:

```
SELECT [...] FROM R
GROUP BY A,B
```

¿Cuál de las siguientes puede aparecer en la posición marcada como [...]?

I. Min(C+D)

II. A,B

III. C,D

a) II solamente

b) I y II solamente

c) I, II y III

d) Ninguno de los anteriores

E4. La siguiente es una secuencia de registros de un log REDO grabado para las transacciones T, U y V:

```
1.  <T, START>
2.  <T, A, 10>
3.  <U, START>
4.  <U, B, 20>
5.  <T, C, 30>
6.  <START CKPT(T,U)>
7.  <U, D, 40>
8.  <U, COMMIT>
9.  <T, E, 50>
10. <V, START>
11. <V, C, 45>
12. <END CKPT>
13. <V, COMMIT>
14. <T, D, 45>
```

Describir brevemente las acciones del recovery manager (cambios a disco solamente), si el sistema bootea después de la caida y descubre este log. Escribir las acciones en el orden que fueron realizadas.

13.1. Resolución

1) Analizamos las instancias, teniendo en cuenta que queremos llegar a un caso en el cual Verdadero implique Falso ($V \rightarrow F$) las primeras dos instancias cumplen que para ellas, la dmv de la izquierda es trivial, por lo tanto es necesariamente cumplida, por lo que el lado izquierdo de la implicación necesariamente es verdadera. Ahora bien ya podemos ver que para el caso a) $A \rightarrow\rightarrow BC$ obviamente se cumple por lo recien mencionado, pero $A \rightarrow\rightarrow B$ no se cumple, pues hay dos tuplas con el mismo A, y por ende debería haber una con B = 11 (de la primer tupla) y C = 22 (de la segunda tupla) cosa que no sucede, por lo tanto sirve de contraejemplo (inclusive, se podría decir que se trata de un contraejemplo trivial). Justamente por ésto mencionado, B) no puede servir de contraejemplo pues cumple con la dmv del lado derecho de la implicación. C y D rápidamente las podemos descartar pues ni siquiera cumplen con la dmv de la izquierda, y el valor de verdad de una implicación cuyo lado implicante es Falso, es

necesariamente verdadero ($v[F \rightarrow ?] = V$).

2) Rápidamente podemos calcular los valores de aquellos que conocemos para ambas tablas:

$$Tam(0) = 5 \times 8 = 40$$

$$Tam(1) = 6 \times 9 = 54$$

$$Tam(2) = 4 \times 5 = 20$$

Ahora bien, tenemos que estimar la cantidad de 3's en la segunda relación. Sabiendo que en 'Otros' tenemos 68 elementos, suponiendo distribución uniforme:

$$Tam(3, S) \approx \frac{Tam(Otros, S)}{V(B, S) - 3} = \frac{68}{21 - 4} = 4$$

Nota: le resto 4 a V(B,S) porque dentro de los 21 elementos diferentes que hay en S, conocemos 4, los cuales NO están dentro de la categoría Otros.

$$Tam(3) = 5 \times 4 = 20$$

Hacemos lo mismo para R, calculando la cantidad de elementos '4':

$$Tam(4, R) \approx \frac{Tam(Otros, R)}{V(B, R) - 4} = \frac{48}{16} = 3$$

La misma explicación vale para haber restado 4 al denominador.

$$Tam(4) = 3 \times 7 = 21$$

Ahora calculamos la cantidad de 'Otros'. Teniendo en cuenta que le sacamos a cada uno la cantidad de elementos estimados anteriormente, la cantidad de otros en R quedaría de 48 - 3 = 45 habiendo 15 'otros' distintos, y 64 'otros' en S, habiendo 16 distintos. La proporción de 'otros' nos va a dar igual a los cálculos hechos antes (4 de cada para S, 3 de cada para R) haciendo la misma cuenta (porque consideramos a todos los casos equiprobables). Por lo tanto:

$$Tam(cada - otros) = 3 \times 4 = 12$$

Esto vale para CADA valor de 'otros' distinto. Ahora falta determinar cuál es la cantidad de otros. En el caso máximo, será el menor número de otros entre R y S (de los restantes que queden). Cabe aclarar que en la realidad podrian ser menos (no más), pero como estamos estimando, no nos importa. En este caso, la varación de otros en R es de 15 y en S de 16, por lo que nos quedamos con 15. Por lo tanto:

$$Tam(otros) = 12 \times 15 = 180$$

El tamaño total será:

$$Tam = Tam(0)+Tam(1)+Tam(2)+Tam(3)+Tam(4)+Tam(otros) = 40+54+20+20+21+180 = 335$$

$$\Rightarrow Tam = 335$$

3) Ésta es fácil: b) la I y II. La 3 no puede ir por el simple hecho que va a andar mal y no estamos poniendo las cosas por lo que agrupamos. Si o si tienen que ir A y B o Min(C+D) (o los dos juntos incluso), por ser una operación de agregación.

4) El sistema va analizando de atrás hacia adelante en busca de los END-CKPT para saber cuales transacciones no es necesario analizar (pues todas las anteriores a su START ya fueron grabadas a disco). En este caso, encuentra un END-CKPT, pero su START incluye a todas las transacciones, por lo que no nos sirvió para evitar tener que rehacer todas las operaciones, pero si antes a éstas hubiera habido alguna transacción W que ya hubiera hecho commit antes del START, entonces no sería necesario rehacer sus acciones. Además se buscan los commits de cada transacción, para saber cuales debe rehacer. En este caso encuentra los commits de U y V. Por lo tanto, irá realizando cada una de las operaciones omitiendo aquellas de T (Se inicia U, se guarda en B el valor 20, se guarda en D el valor 40, Se graba U en disco luego de su commit, Se inicia V, V guarda en C 45, V hace commit, se graba el nuevo valor). Finalmente, se agrega al log una entrada que dice: <ABORT T>, indicando que la transacción T fue abortada (y ninguna de sus modificaciones se ven reflejadas en disco).

14. Final 07/07/2010

E1. Sea $R = (A, B, C, D, E, I)$ $M = \{A \twoheadrightarrow BCD, B \rightarrow AC, C \rightarrow D\}$. ¿Se encuentra en 4FN? Justificar. Si no es así, normalizarlo a 4FN.

E2. Estimar el tamaño de la junta $R(A, B)|x|S(B, C)$ utilizando histogramas.
Asumir V(B,R) = 18 y V(B,S) = 20.
R.B → 0 : 5 filas — 1: 6 filas — 2 : 4 filas — 3 : 5 filas — otros : 42 filas.
S.B → 0 : 10 filas— 1 : 8 filas — 4 : 7 filas — otros : 51 filas.

E3. Sean las Consultas :

```
Q1 = SELECT a from R r1 WHERE EXISTS ( SELECT * from R where a = r1.b );
```

```
Q2 = SELECT a from R WHERE b = ANY ( SELECT a from R );
```

a) Los resultados son iguales.

b) El resultado de Q1 está siempre contenido en el resultado de Q2.

c) El resultado de Q2 está siempre contenido en eñ resultado de Q1.

d) Los resultados son distintos.

Observación : Pueden haber NULL's y filas duplicadas.

E4. Sea el siguiente log REDO :

```
1.<T, START>
2.<T, A, 10>
3.<U, START>
4.<U, B, 20>
5.<T, C, 30>
6.<U, D, 40>
7.<U, COMMIT>
8.<START CKPT ( T )>
9.<T, E, 50>
10.<V, START>
11.<V, C, 45>
12.<END CKPT>
13.<V, COMMIT>
14.<T, D, 45>
15.<T, COMMIT>
```

Describir cuál es el contenido de las variables A, B, C y D si ocurre un error inmediatamente despues de la linea 15 (luego del commit):

a) Antes de la recuperación.

b) Después de la recuperación.

14.1. Resolución

1) Podemos ver que R no está en 4FN porque A no es clave (también se puede ver que por las otras dos dfs tampoco está en FNBC, y si no está en FNBC no puede estar en 4FN). Descomponemos:
$R_1 = (A, B, C, D), M_1 = \{B \to A, B \to C, C- > D\}$ (La dmv no aparece por ser trivial en este esquema).
$R2 = (A, E, I) M_2 = \emptyset$

R_2 se encuentra en 4FN ("no tiene"dmvs ni dfs), mientras que R_1 no, por no cumplir con las condiciones de FNBC ($C \to D$ viola FNBC, y por el axioma de conversión existe $C \to\to D$, y por ser C no-clave, entonces también viola las condiciones de 4FN). Entonces sigo descomponiendo:
$R'_1 = (A, B, C), M'_1 = \{B \to A, B \to C\}$, está en 4FN.
$R3 = (C, D), M_3 = \{C \to D\}$, está en 4FN.

Respuesta final:
$R_1 = (A, B, C) M_1 = \{B \to A, B \to C\}$
$R_2 = (A, E, I) M_2 = \emptyset$
$R_3 = (C, D) M_3 = \{C \to D\}$

2) Muy similar al ejercicio 2 del final del 14/07/2010, con otros valores, resolvemos rápidamente:

$$Tam(0) = 5 \times 10 = 50$$
$$Tam(1) = 6 \times 8 = 48$$

Estimamos la cantidad de 2 y 3 entre los otros de S:

$$Tam(2, 3, S) \approx \frac{Tam(Otros, S)}{V(B, S) - 3} = \frac{51}{20 - 3} = 3$$

Por lo tanto:

$$Tam(2) = 4 \times 3 = 12$$
$$Tam(3) = 5 \times 3 = 15$$

Hacemos la misma estimación para la cantidad de 4's en R:

$$Tam(4, R) \approx \frac{42}{18 - 4} = 3$$

Entonces:

$$Tam(4) = 3 \times 7 = 21$$

Ahora calculamos la cantidad de otros, como ya vimos, la cantidad final de cada otro va a ser la proporción de cada uno distinto de cada uno, multiplicado entre sí (en este caso $3 \times 3 = 9$). Después tenemos que multiplicar por la cantidad de otros que van a haber. En caso de tener el máximo posible (podría no darse si los datos fueran muy distintos) sería agarrando el mínimo que queden de otros entre R y S. En R quedan 17 (sacamos el 4), mientras que en S quedan 18 (sacamos el 2 y el 3), por lo que nos quedamos con 17, entonces el tamaño de otros es:

$$Tam(otros) = 17 \times 9 = 153$$

Por lo tanto, el tamaño total será la suma de todos estos valores:

$$Tam(total) = 50 + 48 + 12 + 15 + 21 + 153 = 299$$

3) La respuesta correcta es la a) Los resultados son iguales (uno de los tantos ejemplos en los que se puede demostrar que algo escrito con any/all se puede llevar a un exists). En el primer caso, nos quedamos con los valores de A que están emparejados con algún B para el cual exista un valor de A igual. En el segundo caso, nos quedamos con los valores de A que estén emparejados con un B, el cual sea igual a algún valor de A, lo cual se puede ver como 'exista algún valor de A igual a él', por lo que puede verse que son lógicamente equivalentes, por lo que los resultados de las consultas serán iguales (ojo, no siempre hay que usar la lógica, porque hay que tener en cuenta la posibilidad de los repetidos en SQL que algunas operaciones omiten y otras no).

4) a) Antes de la recuperación sólo puedo estar seguro que las operaciones que commitearon antes del Start Checkpoint fueron escritas a disco, las demás no puedo asegurar NADA. Por lo tanto, sé que en B está el valor 20(sólo la transacción U la modifica, y ésta hizo commit antes del start ckpt). Sobre el valor de D no tenemos certeza, pues puede valer 40 como tambien 45 (la transacción T la modifica también).

b) Al finalizar la recuperación, dado que todas las transacciones hicieron sus respectivos commits, se reharán sus operaciones (las de V y T, no las de U, porque esta ya fue escrita en disco). Entonces: A = 10, B = 20 (mismo caso al inciso a), C = 45, D = 45 (ahora tenemos la certeza).. De yapa E = 50, aunque no lo pedían.

15. Final 17/02/2010

E1. Sea $R = (A, B, C, D)$. Mostrar una instancia de R que muestre que $B \rightarrow\rightarrow C$ no puede inferirse de $AB \rightarrow C$ y $B \rightarrow\rightarrow D$.

E2. En base a las siguientes estadísticas:

W(A,B)	X(B,C)	Y(C,D)	Z(D,E)
nW = 100	nX = 400	nY = 200	nZ = 300
V(A,W) = 80	V(B,X) = 200	V(C,Y) = 200	V(D,Z) = 200
V(B,W) = 10	V(C,X) = 1	V(D,Y) = 120	V(E,Z) = 80

a) $\sigma_{A=35 \wedge B=5}(W)$

b) $\sigma_{A=35 \vee B=5}(W)$

c) $W \ |x| \ X \ |x| \ Y \ |x| \ Z$

E3. Dada la relación Lista(e,i) donde 'e' (elemento) es un entero, no se repite y no es NULL, e 'i' es la posición en la lista, se pide hacer una consulta SQL para hallar la mediana de "e" (la mediana es un valor que indica que la mitad de los números son mayores que la mediana y la otra mitad es menor). Se puede asumir que la cantidad de tuplas es impar.

E4. Cuales son los valores de los datos A, B, C, D, E, F y G en DISCO teniendo en cuenta que se produce una falla inmediatamente despues de (21) y el siguiente log REDO donde <Tid,Item de Dato, Nuevo valor, Viejo valor>, para los siguientes casos:

a) Antes de la recuperación.

b) Después de la recuperación.

```
01: (START T1)
02: (T1,A,75,20)
03: (START T2)
04: (T1,B,250,20)
05: (T2,C,65,20)
06: (T2,D,99,20)
07:
08: (COMMIT T1)
09: (START T3)
10: (T3,E,55,20)
11:
12: (T2,D,46,20)
13: (COMMIT T2)
14: (START CKPT (T3))
15: (START T4)
16: (T4,F,100,20)
17: (T4,G,111,20)
```

```
18: (COMMIT T3)
19: (END CKPT)
20: (T4,F,150,100)
21: (COMMIT T4)
```

(**Nota**: los datos del log no son fiables porque algunos no fueron copiados, e inventados a mano)

15.1. Resolución

1) Proponemos algo del estilo:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_2	b_1	c_2	d_1
a_2	b_1	c_2	d_2

Vemos que $AB \to C$ se cumple, podemos ver que $B \to\to D$ se cumple, y podemos ver que $B \to\to C$ no se cumple (si aplicamos la definición sobre las tuplas 1 y 3, debería existir una tercer tupla cuyos valores sean $a_2 b_1 c_1 d_1$, lo cual no se cumple, y no se puede cumplir porque a_2 y b_1 tienen que estar con c_2).

2) Ya resuelto, con más o menos esos valores. Ver ej3 del final del 24/07/2013.

3) Ejercicio de la mediana, pero más fácil porque podemos ver la posición:

```
SELECT e
FROM Lista
WHERE i = ((SELECT MAX(i) FROM Lista)+1.0) / 2;
```

Si no estaban ordenados (y pusieron el dato por troll nomás), se resuelve como ya se vió en otros finales.

4) Me suena raro porque dice que es un Log REDO pero tiene la forma de un UNDO/REDO... Quitando eso, es igual (conceptualmente) al ejercicio 4 del final del 14/07/2010.
a) Ya sabemos que T1 y T2 fueron escritos en disco por existir un start-ckpt, entonces A = 20, B = 20, C = 20, D = 20. Del resto no tenemos certezas.

b) Los valores de A, B y C son los mismos al punto anterior, ahora tenemos certeza que D = 20, E = 20, G = 20, pero los registros de modificación de F son inconsistentes (los valores anteriores y posteriores entre registros que lo modifican no concuerdan). Obviamente esto es porque alguien puso valores aleatorios a mano.