

Recuperación Ante Fallas

Base de Datos Junio 2009

1

Mecanismos



- Undo Logging
- Redo Logging
- Undo/Redo Logging
- Con y Sin Checkpoints.

2

Undo Logging



- Mecanismo de recuperación.
- Su objetivo es deshacer transacciones incompletas.

Undo Logging



- <T,x,v>: La transacción T actualizó el dato x y su valor previo era v.
- Politica de logging
 - Los registros del tipo <T,x,v> se escriben a disco antes que el nuevo valor de x se escriba a disco en la base de detes
 - Los registros <Commit T> se escriben a disco después que todos los elementos modificados por T se hayan escrito en disco.

Pasos de Recuperación

- 1. Dividir Tx en:
 - i. Completas:
 - a. <Start T> y <Commit T>
 - b. <Start T> y <Abort T>
 - ii. Incompletas:
 - <Start T> ...
- 2. Para toda Tx Incompleta
 - Recorrer log en sentido inverso y para cada
 T, x, v> se deshace la modificación.
 - ii. Agregar <Abort Tx>

5

Ejemplo



# Paso	Registro
1	<start t=""></start>
2	<t,a,8></t,a,8>
3	<t,a,16></t,a,16>
4	<t,b,8></t,b,8>

- Caso 1: Falla! Ultimo registro <T, B, 8> (4)
- Transacciones a deshacer:
- T
- Cambios en la base de datos:
 - B = 8 (P.4)
 - A = 16 (P.3)
 - A = 8 (P.2)
- Cambios en el log: Agregar < Abort T>

6

Ejemplo

<T,A,8>

<T,A,16>

<T,B,8>



- Caso 2: Falla! Ultimo
 registro <Commit T> (5)
 - No hacemos nada ©.

Undo con Checkpoints - Quiescente



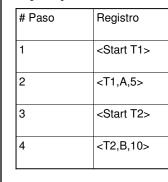
- Sin checkpoints Problemas
 - Hay que recorrer todo el log siempre! 🕾
- Introducimos checkpoints para hacer más eficiente la recuperación.

Politica para Checkpoint Quiescente

- 1. Dejar de aceptar nuevas transacciones.
- Esperar a que todas las transacciones activas o bien comiteen o aborten.
- 3. Escribir un registro <CKPT> en el log y luego efectuar un flush.
- 4. Aceptar nuevas transacciones.

(Quiescente: Que está quieto pudiendo tener movimiento propio)

Ejemplo





- Se decide introducir un checkpoint.
- Se "detiene" la base de datos (ie, no se aceptan nuevas transacciones).
- Se espera que terminen T1 y T2.

Ejemplo (Continuación)



- El registro CKPT recién puede introducirse cuando terminan T1
- Supongamos una falla luego del paso 12.
- · Acciones de Recuperación
 - Debo considerar únicamente hasta el paso 9.
 - · Transacciones a deshacer: T3.
 - Cambios en la base de datos:
 - F:=29 (paso 12)
 - R:=22 (paso 11)
 - Cambios en el log: <Abort T3.>

Undo con Checkpoints-No Quiescente



- Politica para introducir los checkpoints
 - Escribir un registro <Start CKPT(T1,T2,...,Tk)> en el log. T1,T2,...,Tk son las transacciones activas (aquellas con <START T> y sin <Commit T>) al momento de introducir el checkpoint.
 - 2. Esperar a que todas las transacciones T1,T2,...,Tk terminen (ya sea abortando o comiteando).
 - Escribir el registro <End CKPT> en el log y efectuar un flush.

Recuperación

- Si encontramos un <End CKPT>, sabemos por la segunda condición que todas las transacciones que estaban activas al momento del Start CKPT terminaron.
- Si no encontramos el registro <End CKPT>, existe al menos una transacción activa al momento del Start CKPT todavía no comiteó, por lo que debemos deshacer todas sus acciones, hasta encontrar su registro <Start T>.

13

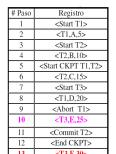
Ejemplo



# Paso	Registro
1	<start t1=""></start>
2	<t1,a,5></t1,a,5>
3	<start t2=""></start>
4	<t2,b,10></t2,b,10>
5	<start ckpt="" t1,t2:<="" th=""></start>
6	<t2,c,15></t2,c,15>
7	<start t3=""></start>
8	<t1,d,20></t1,d,20>
9	<abort t1=""></abort>
10	<t3,e,25></t3,e,25>
- 11	<commit t2=""></commit>
12	<end ckpt=""></end>
13	<t3,f,30></t3,f,30>
	·

- Caso: Falla! Ultimo registro <T3, F, 30>
- Transaccion a deshacer:
- T3
- · Cambios en DB
 - F=30 (P.13)
 - E=25 (P.10)
- · Cambios en el log
 - Agrego <Abort T3> 14

Ejemplo: Caso 2



- Caso: Falla! Ultimo registro <T3, E, 25>
- Transaccion a deshacer:
- T2 y T3
- · Cambios en DB
 - E=25 (P.10)
 - C:=15(P. 6)
 - B:=10(P.4)
- · Cambios en el log
 - <Abort T3> <Abort T2>

15

Redo Logging



- <T,x,v>: La transacción T actualizó el dato x y su valor actual es v.
- Politica de logging
 - Los registros del tipo <T,x,v> se escriben a disco antes que el nuevo valor de x se escriba a disco en la base de datos.
 - Los registros <Commit T> se escriben a disco antes que todos los elementos modificados por T se hayan escrito en disco.

Pasos de Recuperación

- 1. Dividir Tx en Completas e Incompletas.
- 2. Para toda Tx con Start y Commit
 - Recorrer log de ppio a fin y para cada <T, x, v> se efectúa la modificación.
- 3. Para toda Tx incompleta
 - Agregar <Abort Tx>

17

Ejemplo

Registro

<Start T>

<T,A,16>

<T,A,32>

<T,B,16>

<Commit '

Paso

2

3

4



- registro <T, B, 8> (4)
- · Transacciones a rehacer:
- Ninguna
- · Cambios en el log:
 - Agregar <Abort T>
 - ¿Cuales son las acciones si el último reg fuera el #5.?

18

Redo con checkpoint



- Politica para introducir los checkpoints
 - Escribir un registro <Start CKPT(T1,T2,...,Tk)> en el log. T1,T2,...,Tk son las transacciones activas (aquellas con <START T> y sin <Commit T>) al momento de introducir el checkpoint.
 - Esperar a que todas las modificaciones realizadas por transacciones ya comiteadas al momento de introducir el Start CKPT sean escritas a disco.
 - Escribir el registro <End CKPT> en el log y efectuar un flush.

Pasos de Recuperación



- 1. Para toda Tx con Start y Commit
 - i. Si existe <End CKPT>
 - a. Ignoramos las Tx comiteadas antes del <Start CKPT(T1,T2,...,Tk)>
 - b. Empezar desde el start más antiguo de las Ti
 - ii. Si NO existe <End CKPT>: retroceder hasta Start Ckpt Anterior.
- 2. Para toda Tx incompleta
 - Agregar < Abort Tx>

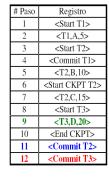
Ejemplo

# Paso	Registro
1	<start t1=""></start>
2	<t1,a,5></t1,a,5>
3	<start t2=""></start>
4	<commit t1=""></commit>
5	<t2,b,10></t2,b,10>
6	<start ckpt="" t2=""></start>
7	<t2,c,15></t2,c,15>
8	<start t3=""></start>
9	<t3,d,20></t3,d,20>
10	<end ckpt=""></end>
- 11	<commit t2=""></commit>
12	<commit t3=""></commit>

- Caso: Falla! Ultimo registro
 Commit T2>
- Transaccion a rehacer:
- T2 (ignoro T1).
- Comienzo: desde el start de T2 en el paso 3.
 - B:=10;C:=15.
- · Cambios en el log
 - <Abort T3>

21

Ejemplo



- Caso: Falla! Ultimo registro
 T3, D, 20>
- Transaccion a rehacer:
- T1
- · Cambios en DB
 - A=5 (P.2)
- · Cambios en el log
 - <Abort T3>
 - <Abort T2>

Undo/Redo sin Checkpoints



- <T, x, v, w>: La transacción T actualizó el dato x, su valor previo era v y su valor actual
- Politica de logging
 - Los registros del tipo <T,x,v, w> se escriben a disco antes que el nuevo valor de x se escriba a disco en la base de datos.

Pasos de Recuperación



- Deshacer transacciones incompletas (Undo logging)
- 2. Rehacer transacciones con start y commit (Redo logging)

Nota: Ignoramos Tx con start y abort.

Undo/Redo Checkpoint-No Quiescente

- Politica para introducir los checkpoints
 - Escribir un registro <Start CKPT(T1,T2,...,Tk)> en el log. T1,T2,...,Tk son las transacciones activas (aquellas con <START T> y sin <Commit T>) al momento de introducir el checkpoint.
 - Esperar a que todas las modificaciones realizadas por transacciones comiteadas o no, al momento de introducir el Start CKPT sean escritas a disco.
 - Escribir el registro <End CKPT> en el log y efectuar un fluch

25

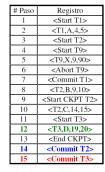
Pasos de recuperación



- Deshacer transacciones incompletas (Undo logging)
 - Debemos retroceder hasta el start más antiguo de ellas.
- 2. Rehacer transacciones con start y commit (Redo logging)
 - 1. Si existe <END CKPT>
 - Rehago desde el último <START CKPT> en adelante.
 - 2. Si NO existe <END CKPT>
 - 1. Idem Redo.

20

Ejemplo





- Caso: Falla! Ultimo registro <Commit T2>
- Transacciones a deshacer:
- T3
- Transacciones a rehacer:
- T1, T2
- Cambios en DB
 - [deshacer] D=19 (P.12)
 - [rehacer] C=15 (P.10)
- Cambios en el log: <Abort T3>