Recuperación Ante Fallas

Base de Datos
Junio 2009

Mecanismos

- Undo Logging
- Redo Logging
- Undo/Redo Logging

- Con y Sin Checkpoints.

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 datos.*
  - 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
   i. Recorrer log en sentido inverso y para cada <T, x, v> se deshace la modificación.
   ii. Agregar <Abort Tx>

Ejemplo

# Paso	Registro
1	<Start T>
2	<T,A,8>
3	<T,A,16>
4	<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>

Ejemplo

# Paso	Registro
1	<Start T>
2	<T,A,8>
3	<T,A,16>
4	<T,B,8>
5	<Commit T>

- 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.
2. 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

# Paso	Registro
1	<Start T1>
2	<T1,A,5>
3	<Start T2>
4	<T2,B,10>

- 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)

# Paso	Registro
1	<Start T1>
2	<T1,A,5>
3	<Start T2>
4	<T2,B,10>
5	<T2,C,15>
6	<T1,D,20>
7	<Commit T1>
8	<Abort T2>
9	**<CKPT>**
10	<Start T3>
11	<T3,R,22>
12	<T3,F,29>

- El registro CKPT recién puede introducirse cuando terminan T1 y T2.
- 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
  1. 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).
  3. Escribir el registro <End CKPT> en el log y efectuar un flush.

Recuperación

1. Si encontramos un <End CKPT>, sabemos por la segunda condición que todas las transacciones que estaban activas al momento del Start CKPT terminaron.

2. 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>.

Ejemplo

# Paso	Registro
1	<Start T1>
2	<T1,A,5>
3	<Start T2>
4	<T2,B,10>
5	<Start CKPT T1,T2>
6	<T2,C,15>
7	<Start T3>
8	<T1,D,20>
9	<Abort T1>
10	<T3,E,25>
11	<Commit T2>
12	<End CKPT>
13	<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>

Ejemplo: Caso 2

# Paso	Registro
1	<Start T1>
2	<T1,A,5>
3	<Start T2>
4	<T2,B,10>
5	<Start CKPT T1,T2>
6	<T2,C,15>
7	<Start T3>
8	<T1,D,20>
9	<Abort T1>
10	<T3,E,25>
11	<Commit T2>
12	<End CKPT>
13	<T3,F,30>

- 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>

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>

Ejemplo

# Paso	Registro
1	<Start T>
2	<T,A,16>
3	<T,A,32>
4	<T,B,16>
5	<Commit T>

- Caso 1: Falla! Ultimo 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.?

Redo con checkpoint

- Politica para introducir los checkpoints
  1. 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 modificaciones realizadas por transacciones ya comiteadas al momento de introducir el Start CKPT sean escritas a disco.***
  3. 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
   1. Agregar <Abort Tx>

Ejemplo

# Paso	Registro
1	<Start T1>
2	<T1,A,5>
3	<Start T2>
4	<Commit T1>
5	<T2,B,10>
6	<Start CKPT T2>
7	<T2,C,15>
8	<Start T3>
9	<T3,D,20>
10	<End CKPT>
11	<Commit T2>
12	<Commit T3>

- 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>

Ejemplo

# Paso	Registro
1	<Start T1>
2	<T1,A,5>
3	<Start T2>
4	<Commit T1>
5	<T2,B,10>
6	<Start CKPT T2>
7	<T2,C,15>
8	<Start T3>
9	<T3,D,20>
10	<End CKPT>
11	<Commit T2>
12	<Commit T3>

- 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 w.*
- *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

1. 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
  1. 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 modificaciones realizadas por transacciones comiteadas o no, al momento de introducir el Start CKPT sean escritas a disco.***
  3. Escribir el registro <End CKPT> en el log y efectuar un flush.

Pasos de recuperación

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

Ejemplo

# Paso	Registro
1	<Start T1>
2	<T1,A,4,5>
3	<Start T2>
4	<Start T9>
5	<T9,X,9,90>
6	<Abort T9>
7	<Commit T1>
8	<T2,B,9,10>
9	<Start CKPT T2>
10	<T2,C,14,15>
11	<Start T3>
12	<T3,D,19,20>
13	<End CKPT>
14	<Commit T2>
15	<Commit T3>

- 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>