Recuperación ante fallas
Clase Práctica Checkpointing

Rosana Matuk

Departamento de Computación - Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Base de Datos
2do. Cuatrimestre 2014

Esquema General

Esquema General

Se cayó el sistema!!! Y ahora qué hacemos?

Recuperación ante fallas

Objetivo

Llevar la base de datos a un estado consistente, luego de la ocurrencia de una falla del sistema ("crash").

Técnicas

Basadas en el uso del Log , el cual registra la historia de los cambios sobre la base de datos.

Esquema General

Esquema General

Tres Políticas

Undo

Deshace las transacciones incompletas

Redo

Rehace las transacciones que hicieron COMMIT

Undo/Redo

Deshace las transacciones incompletas y rehace las que hicieron COMMIT

Política UNDO

- **Recovery**: **Deshace las transacciones incompletas.** Para cada transacción incompleta T, escribe un `<ABORT T>` al log y flush del log.
- **Log:** `<T,X,ValorAnterior>`
- **Reglas:**
  1. 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.
  2. El registro `<Commit T>` se escribe a disco después que todos los elementos modificados por *T* se hayan escrito en disco.

Política REDO

- **Recovery**: **Rehace las transacciones que hicieron COMMIT.** Para cada transacción incompleta T, escribe un `<ABORT T>` al log y flush del log.
- **Log:** `<T,X,ValorNuevo>`
- **Reglas:**
  1. 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.
  2. El registro `<Commit T>` se escribe a disco antes que cualquier elemento modificado por *T* se haya escrito en disco.

Política UNDO/REDO

- **Recovery**: **Deshace todas las transacciones incompletas** a partir de la más reciente y **rehace todas las transacciones comiteadas** a partir de la más antigua. Para cada transacción incompleta T, escribe un `<ABORT T>` al log y flush del log.
- **Log:** `<T,X,ValorAnterior,ValorNuevo>`
- **Reglas:**
  1. 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.

Esquema General

2 Políticas y Checkpointing

- Políticas de Recuperación
- Checkpointing

Checkpointing

Objetivo

Limitar la cantidad de registros del Log que deben ser examinados en la recuperación.

Tipos

1. **Quiescente**: No aceptan nuevas transacciones (el sistema queda "inactivo" durante el checkpoint).
2. **No Quiescente**: Aceptan nuevas transacciones durante el checkpoint.

Esquema General

Esquema General

UNDO / Checkpoint Quiescente

Etapas:

1. Dejar de aceptar nuevas transacciones.
2. Esperar a que todas las transacciones activas (aquellas con `<START T>` y sin commit ni abort) comiteen o aborten.
3. Escribir un `<CKPT>` en el log y luego efectuar un flush.
4. Aceptar nuevas transacciones.

Recovery: Se lee el Log a partir del último registro y hasta el punto de checkpoint. Sabemos que hasta ahí todas las transacciones terminaron y sus cambios fueron guardados en disco. Aplicar la política del UNDO.

Ejemplo (UNDO / Checkpoint Quiescente): enunciado

#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,E,25>

Preguntas:

1. Se introduce un Checkpoint Quiescente en el Paso #4. Identifique las etapas del Checkpoint en la tabla del log.
2. Se produjo un Crash, luego del paso #11. Describa el recovery.

Ejemplo (UNDO / Checkpoint Quiescente): resolución

#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,E,25>

Paso#4: Se decide introducir un CKPT.
No se aceptan nuevas transacciones.
Deben terminar T1 y T2, las transacciones activas.

Paso#8: T1 y T2 terminaron.
Paso#9: Se agrega el registro <CKPT>.
Paso#10: Se aceptan nuevas transacciones.

Recovery.- ***Crash luego del paso 11:*** Recorremos desde el final hasta el último CKPT, en el paso 9. Se deshace T3 (E=25). Se agrega un registro <ABORT T3> y flush del Log.

Esquema General

3 Análisis de Casos

- Checkpoint Quiescente
- Checkpoint No Quiescente

UNDO / Checkpoint NO Quiescente

Etapas:

1. Escribir `<Start CKPT(T1,T2,...,Tk)>` en el log, y efectuar un flush. T1,T2,...,Tk son las transacciones activas al momento de introducir el checkpoint.
2. Esperar a que todas las transacciones T1,T2,...,Tk terminen (ya sea abortando o comiteando), pero sin prohibir que empiecen otras transacciones.
3. Escribir `<End CKPT>` en el log y efectuar un flush.

Recovery: Leer el Log desde el último registro, aplicando la política UNDO.

1. Encuentro un `<END CKPT>`, debo leer no más allá del `<START CKPT>`.
2. Encuentro un `<START CKPT>` (no hay un `<END CKPT>` debido a un crash). Debo leer el Log hasta el START más antiguo de las transacciones de la lista del start ckpt que quedaron incompletas.

Ejemplo (UNDO / Checkpoint NO Quiescente): enunciado

#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>

Preguntas:

1. Se introduce un Checkpoint No Quiescente. Identifique las etapas del Checkpoint en la tabla del log.
2. Se produjo un Crash, y el último registro es el #13. Describa el recovery. Idem, si el último registro fuese el #8.

Ejemplo (UNDO / Checkpoint NO Quiescente): resolución

#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>

Paso#5: Se decide introducir un CKPT.
Las transacciones T1 y T2 están activas.
Paso#7: Se aceptan nuevas transacciones.

Paso#11: T1 y T2 terminaron.
Paso#12: Se agrega el registro <END CKPT>

Recovery:

- ***Crash y último registro es el #13:*** Se lee hasta el Start CKPT (#5). Se deshace T3 (F=30, E=25). Se agrega un registro <ABORT T3> y flush del Log.
- ***Crash y último registro es el #8:*** Transacciones incompletas: T1,T2 y T3. Las deshago (D=20,C=15,B=10,A=5). Agrego registros <ABORT T3> <ABORT T2> <ABORT T1> y flush del Log.

REDO / Checkpoint NO Quiescente

Etapas:

1. Escribir `<Start CKPT(T1,T2,...,Tk)>` en el log, y efectuar un flush. T1,T2,...,Tk son las transacciones activas (aquellas con `<START T>` y sin commit ni abort) al momento de introducir el checkpoint.
2. Esperar a que todas las modificaciones realizadas en los buffers por transacciones **ya comiteadas** al momento del Start CKPT sean escritas a disco.
3. Escribir `<End CKPT>` en el log y efectuar un flush.

Recovery: Se aplica la política REDO. Si leyendo el log desde el último registro,

1. Encuentro un `<END CKPT>`. Indica que las transacciones que hicieron COMMIT antes del START CKPT tienen sus cambios en disco, luego las ignoro. Rehacer las transacciones que hicieron COMMIT, y que comenzaron luego del START CKPT, o que estuvieron activas al momento del START CKPT, desde el START más antiguo de dichas transacciones.
2. Encuentro un `<START CKPT...>` (no hay un `<END CKPT>` debido a un crash). Ubicar el `<END CKPT>` anterior y su correspondiente `<Start CKPT(S1,S2,...,Sk)>` y rehacer todas las transacciones comiteadas que comenzaron a partir de ese START CKPT o están entre las S_i, desde el START más antiguo de dichas transacciones.

Ejemplo (REDO / Checkpoint NO Quiescente): enunciado

#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>

Preguntas:

1. Se introduce un Checkpoint No Quiescente. Identifique las etapas del Checkpoint en la tabla del log.
2. Se produjo un Crash, y el último registro es el #12. Describa el recovery. Idem, si el último registro fuese el #7.

Ejemplo (REDO / Checkpoint NO Quiescente): resolución

#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>

Paso#6: Se decide introducir un CKPT. Las transacción T2 está activa.

Paso#8: Se aceptan nuevas transacciones.

Paso#10: <END CKPT> indica que las modificaciones de T1 ya están en disco.

Recovery:

- **Crash y el último registro es el #12:** se rehacen T2 y T3 comenzando desde el registro #3 (B=10, C=15, D=20).
- **Crash y el último registro es el #7:** Se rehace T1 comenzando desde el registro #1 (A=5). Agrego registro <ABORT T2> y flush del Log.

UNDO/REDO con Checkpoint NO Quiescente

Etapas:

1. Escribir un `<Start CKPT(T1,T2,...,Tk)>` en el log, y efectuar un flush. T1,T2,...,Tk son las transacciones activas (aquellas con `<START T>` y sin commit ni abort) al momento de introducir el checkpoint.
2. Escribir a disco todos los buffers "sucios" (contienen elementos cambiados que aún no fueron escritos a disco) al momento del START CKPT. Escribir a disco **todos** los buffers sucios, no sólo los que hayan sido modificados por transacciones comiteadas al momento del START CKPT.
3. Escribir `<End CKPT>` en el log y efectuar un flush.

Recovery: Se aplica la política UNDO/REDO.

1. *Transacciones incompletas*: para deshacerlas deberemos retroceder hasta el start más antiguo de ellas. Agregar registro ABORT al log para cada transacción incompleta, y hacer flush del log.
2. *Transacciones comiteadas*: Si leyendo desde el último registro del log, encontramos un registro `<End CKPT>` sólo será necesario rehacer las acciones efectuadas desde el correspondiente registro Start CKPT en adelante.

Ejemplo (UNDO/REDO con Checkpoint NO Quiescente): enunciado

#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>

Preguntas:

1. Se introduce un Checkpoint No Quiescente. Identifique las etapas del Checkpoint en la tabla del log.
2. Se produjo un Crash, y el último registro es el #14. Describa el recovery. Idem, si el último registro fuese el #10.

Ejemplo (UNDO/REDO con Checkpoint NO Quiescente): solución

#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>

Paso#9: Se decide introducir un CKPT.
Las transacción T2 está activa.

Paso#11: Se aceptan nuevas transacciones.

Paso#13: <END CKPT> indica que todos los cambios antes del START CKPT están en disco.

Recovery:

- **Crash y el último registro es el #14:** se deshace T3 (D=19). Se agrega registro <ABORT T3> y flush del Log. Se rehace T2 (C=15), se lee a partir del START CKPT. Se ignora T1.
- **Crash y último registro es el #10:** Se deshace T2 (C=14, B=9). Se agrega registro <ABORT T2> y flush del Log. Se rehace T1, leyendo desde el registro 1 (A=5).

Bibliografía

Y llegamos al <COMMIT> de esta clase!!!

Referencia

Héctor García-Molina, Jeffrey D. Ullman, Jennifer Widom, "Database Systems: The Complete Book", Segunda Edición, Prentice Hall, 2008 *(capítulo 17, secciones 1-4)*.