

Ejercicios de Logging (libro de Ullman, cap. 17)

Política Undo con checkpoint quiescente

```
<START  $T_1$ >  
< $T_1$ , A, 5>  
<START  $T_2$ >  
< $T_2$ , B, 10>  
< $T_2$ , C, 15>  
< $T_1$ , D, 20>  
<COMMIT  $T_1$ >  
<COMMIT  $T_2$ >  
<CKPT>  
<START  $T_3$ >  
< $T_3$ , E, 25>  
< $T_3$ , F, 30>
```

Figure 17.4: An undo log

1. ¿Cómo se hace el recovery si la base falla en la última operación?
2. ¿Qué pasa con T_1 y T_2 si falla entre $\text{<COMMIT } T_1\text{>}$ y $\text{<COMMIT } T_2\text{>}$?

Política Undo con checkpoint quiescente

Respuestas:

1. Se deshacen los cambios de T_3
2. Hay que deshacer sólo los cambios de T_2 porque T_1 hizo el commit antes, por lo tanto estamos seguros de que sus cambios se grabaron.

Política Undo con checkpoint no quiescente

```
<START  $T_1$ >  
< $T_1$ , A, 5>  
<START  $T_2$ >  
< $T_2$ , B, 10>  
<START CKPT ( $T_1$ ,  $T_2$ )>  
< $T_2$ , C, 15>  
<START  $T_3$ >  
< $T_1$ , D, 20>  
<COMMIT  $T_1$ >  
< $T_3$ , E, 25>  
<COMMIT  $T_2$ >  
<END CKPT>  
< $T_3$ , F, 30>
```

Figure 17.5: An undo log using nonquiescent checkpointing

1. ¿Qué pasa si la base falla inmediatamente después de la última operación?
2. ¿Y si falla justo antes del $\text{<COMMIT } T_2\text{>}$?

Política Undo con checkpoint quiescente

Respuestas:

1. Hay que revertir sólo los cambios de T_3 porque al encontrar el registro $\langle \text{END CKPT} \rangle$ sabemos que todas las transacciones incompletas empezaron después del $\langle \text{START CKPT} \rangle$ (y T_3 es la única en esa situación)
2. Revisando el log de atrás para adelante nos damos cuenta de que T_3 y T_2 están incompletas. Por lo tanto hay que revertirlas. Por otro lado, al identificar el $\langle \text{START CKPT}(T_1, T_2) \rangle$, sabemos que no hay otras transacciones incompletas (porque T_1 pudo commitear antes del crash).

Política Redo con checkpoint no quiescente

```
<START  $T_1$ >  
< $T_1$ , A, 5>  
<START  $T_2$ >  
<COMMIT  $T_1$ >  
< $T_2$ , B, 10>  
<START CKPT ( $T_2$ )>  
< $T_2$ , C, 15>  
<START  $T_3$ >  
< $T_3$ , D, 20>  
<END CKPT>  
<COMMIT  $T_2$ >  
<COMMIT  $T_3$ >
```

Figure 17.8: A redo log

1. ¿Qué pasa si la base falla entre el $\langle \text{COMMIT } T_2 \rangle$ y el $\langle \text{COMMIT } T_3 \rangle$?
2. ¿Y si falla inmediatamente antes del $\langle \text{END CKPT} \rangle$?

Política Redo con checkpoint no quiescente

Respuestas:

1. Como encontramos un $\langle \text{END CKPT} \rangle$, sabemos que todas las transacciones commiteadas antes del correspondiente $\langle \text{START CKPT} \rangle$ fueron grabadas. Hay que concentrarse en las transacciones que estaban pendientes en el momento del $\langle \text{START CKPT} \rangle$ (T_2) y las que comenzaron después (T_3). T_2 llegó a commitear por lo tanto tenemos que rehacer sus cambios. Hay que abortar T_3 porque no llegó a commitear.
2. Como encontramos un $\langle \text{START CKPT} \rangle$ no podemos determinar si las transacciones commiteadas antes del $\langle \text{START CKPT} \rangle$ fueron grabadas, por lo tanto tenemos que retroceder hasta el anterior $\langle \text{END CKPT} \rangle$ y encontrar su correspondiente $\langle \text{START CKPT}(S_1, \dots, S_m) \rangle$ y tenemos que rehacer las transacciones S_i que hayan commiteado y las posteriores a ese $\langle \text{START} \rangle$ que hayan commiteado. Las que no hayan commiteado deben abortarse.

Política Undo/Redo con checkpoint no quiescente

```
<START T1>  
<T1, A, 4, 5>  
<START T2>  
<COMMIT T1>  
<T2, B, 9, 10>  
<START CKPT (T2)>  
<T2, C, 14, 15>  
<START T3>  
<T3, D, 19, 20>  
<END CKPT>  
<COMMIT T2>  
<COMMIT T3>
```

Figure 17.10: An undo/redo log

1. ¿Qué pasa si la base falla inmediatamente después del final del log?
2. ¿Y si falla entre $\text{<COMMIT } T_2\text{>}$ y $\text{<COMMIT } T_3\text{>}$?

Política Redo con checkpoint no quiescente

Respuestas:

1. Hay que rehacer los cambios de T_2 y T_3 porque commitearon después del <START CKPT>. En el caso de T_2 basta rehacer a partir del <START CKPT> porque los cambios anteriores se grabaron. Como T_1 commiteo antes del <START CKPT> estamos seguros de que sus cambios se grabaron.
2. T_2 se marca como commiteada y T_3 como incompleta. Hay que deshacer T_3 completamente y marcarla como abortada. Y hay que rehacer T_2 a partir del <START CKPT> (porque sus cambios anteriores se grabaron)