Ayudantía 12 Bases de Datos

Paula Verástegui - Olivia Llanos

Schedules

Secuencia de operaciones realizadas por múltiples transacciones de forma concurrente.

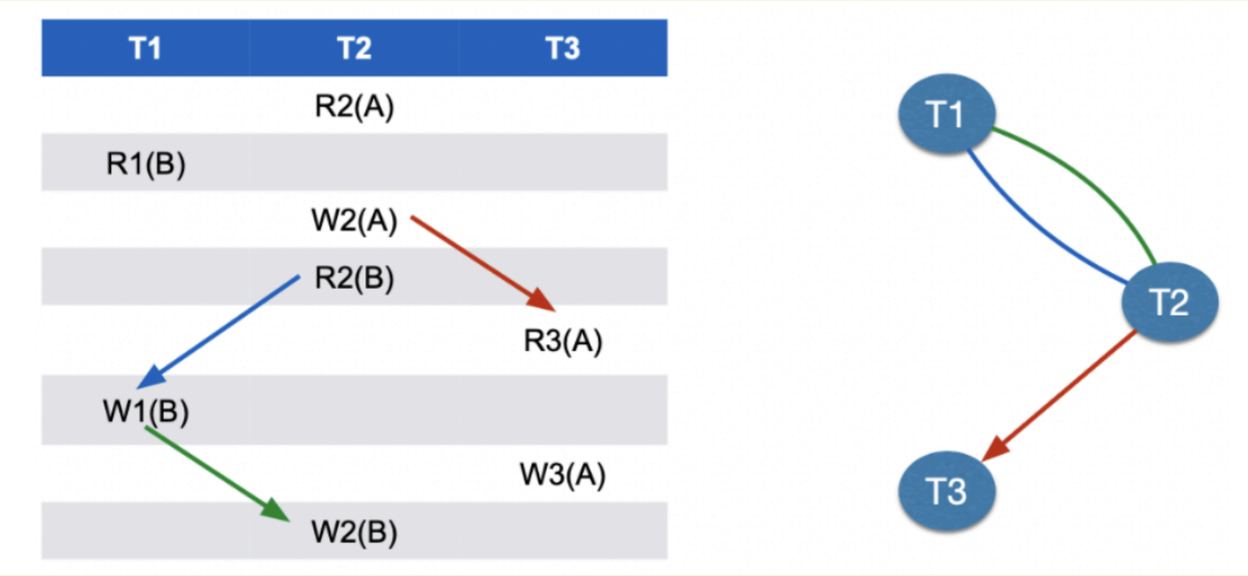
- Está compuesto por **operaciones** de **múltiples transacciones** que pueden ejecutarse de manera concurrente.
- Define el **orden** en el cual las operaciones de las transacciones son ejecutadas.
- Puede ser **Serializable** o **No Serializable**: depende de si la ejecución produce el mismo resultado que alguna secuencia serial (no concurrente) equivalente de las mismas transacciones.

Schedules

- **Schedule Serial:** no hay intercalación entre las acciones.
- **Schedule Serializable:** si existe algún schedule S’ serial con las mismas transacciones, tal que el resultado de S y S’ es el mismo.
- **Schedule No Serializable:** produce un resultado diferente que cualquier secuencia serial equivalente de las mismas transacciones.
- **Schedule Conflict Serializable:** un schedule es conflict serializable si puedo
- transformarlo a uno serial usando permutaciones.

Grafos de procedencia

Representación gráfica que muestra las dependencias de orden entre las operaciones de diferentes transacciones.



*** Un schedule es conflict serializable si solo si el grafo es acíclico (en este caso NO lo es).

Schedules - Ejercicio

Considere el Schedule del cuadro 1. Indique para cada valor de X (puede ser R o W) si el schedule es conflict serializable o no. Explique por qué:

T1	T2	T3	T4
R(a)			
	W(a)		
	R(b)		
		W(b)	
		W(c)	
			W(c)
			R(d)
X(d)			

Schedules - Ejercicio

Considere el Schedule del cuadro 1. Indique para cada valor de X (puede ser R o W) si el schedule es conflict serializable o no. Explique por qué:

T1	T2	T3	T4
R(a)			
	W(a)		
	R(b)		
		W(b)	
		W(c)	
			W(c)
			R(d)
X(d)			

Schedules – Ejercicio

Considere el Schedule del cuadro 1. Indique para cada valor de X (puede ser R o W) si el schedule es conflict serializable o no. Explique por qué:

T1	T2	T3	T4
R(a)			
	W(a)		
	R(b)		
		W(b)	
		W(c)	
			W(c)
			R(d)
X(d)			

X = R: T1 lee(d) y T4 también lee(d).

Como ambas operaciones son lecturas (**R - R**), no hay conflicto. No se sobre escribe ningún valor y no se afecta el resultado obtenido por las transacciones.

T1 → T2 → T3 → T4

Schedules - Ejercicio

Considere el Schedule del cuadro 1. Indique para cada valor de X (puede ser R o W) si el schedule es conflict serializable o no. Explique por qué:

T1	T2	T3	T4
R(a)			
	W(a)		
	R(b)		
		W(b)	
		W(c)	
			W(c)
			R(d)
X(d)			

X = W: T1 escribe(d) y T4 lee(d).

Esto es una combinación **W - R**, sí hay conflicto, se genera un ciclo en el grafo de procedencia:

T1 → **T2** → **T3** → **T4** → **T1**

Strict 2PL

Control de concurrencia. Está basado en la utilización de locks.

- **Shared Locks** (**Read**)**:** lectura de objetos realizado por una transacción. Permiten que varias transacciones lean los mismos datos simultáneamente.

- **Exclusive Locks** (**Write**)**:** escritura sobre objetos realizado por una transacción. Permiten que solo una transacción lea o modifique datos a la vez.

Si una transacción T quiere leer un objeto (R), pide un shared lock (Read) sobre el objeto. Exclusive lock es para Write (W).

Strict 2PL

La utilización de los locks sobre las schedule se basa en 2 reglas.

- **Regla 1:** Una transacción que pide un lock, se suspende hasta que el lock es otorgado. Si una T mantiene un exclusive lock de un objeto, ninguna otra transacción puede mantener un shared o exclusive lock sobre ese objeto y tampoco puede obtenerlo (el exclusive lock) si ya hay un lock sobre el objeto.

- **Regla 2:** Cuando la transacción se completa, libera todos los locks que mantenía.

Schedules - Ejercicio

Considere el Schedule del cuadro 2. Diga como lo reordenaría/ejecutaría Strict-2PL. ¿Cambiaría el orden de algunas operaciones/transacciones? Para esto considera como se asignan shared/exclusive locks.

T1	T2	T3
R(a)		
	R(b)	
		W(a)
		W(c)
R(c)		
	R(c)	

Ejercicio - Solución

Considere el Schedule del cuadro 2. Diga como lo reordenaría/ejecutaría Strict-2PL. ¿Cambiaría el orden de algunas operaciones/transacciones? Para esto considera como se asignan shared/exclusive locks.

T1	T2	T3
R(a)		
	R(b)	
		W(a)
		W(c)
R(c)		
	R(c)	

- T1 – shared lock (A)
- T2 – shared lock (B)
- T3 – pide exclusive (A), sistema dice espera que termine T1
- T3 – está congelada, operación W(c) está igual en espera
- T1 – shared lock (C) – al ejecutar se liberan locks de T1
- T2 – shared lock (C) – al ejecutar se liberan los locks de T2
- Ahora va T3

Ejercicio – Otra Solución Válida

Considere el Schedule del cuadro 2. Diga como lo reordenaría/ejecutaría Strict-2PL. ¿Cambiaría el orden de algunas operaciones/transacciones? Para esto considera como se asignan shared/exclusive locks.

T1	T2	T3
R(a)		
	R(b)	
		W(a)
		W(c)
R(c)		
	R(c)	

- T1 – shared lock (A)
- T2 – shared lock (B)
- T3 – pide exclusive (A), sistema dice espera que termine T1
- T3 – está congelada, operación W(c) está igual en espera
- T1 – shared lock (C) – al ejecutar se liberan locks de T1
- T3 - exclusive lock (A)
- T3 - exclusive lock (C) - al ejecutar se liberan todos los locks de T3
- T2 – shared lock (C) – al ejecutar se liberan los locks de T2

Undo Loggin

Regla 1:

Si T modifica X, todos log <T, X, T> deben ser escritos antes que el valor X sea escrito en disco.

Regla 2:

Si **T** hace COMMIT, el log <**COMMIT T**> debe ser escrito justo después de que todos los datos modificados por T estén almacenado en disco.

1. T cambia el valor del X (t es el valor antiguo)
2. Generar el log <T, X, t>
3. Escribir los logs <**T,** X, t> al disco
4. Escribir valor nuevo de X a disco.
5. Escribir <COMMIT **T**>

Undo Loggin Recovery

¿Qué hacer cuando hay falla?

- Si leo <COMMIT T>, marco **T** como realizada
- Si leo <ABORT T>, marco **T** como realizada
- Si leo <T, X, t>, debo restituir **X := t** en disco, si no fue realizada.
- Si leo <START T>, lo ignoro

Esto solo se realiza dentro de la sección donde hay falla, no en toda la ejecución!

Undo Loggin

Recovery con checkpoints:

- Si encontramos un <END CKPT>, hacemos undo de todo lo que haya empezadodespués del inicio del checkpoint <START CKPT>.
- Si encontramos un <START CKPT (T1 ...TN)> sin su <END CKPT>, debemos analizar el log desde el inicio de la transacción más antigua entre T1 ,...,TN.

Undo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Undo”. Suponiendo que la política de recovery es la de Undo Logging, indique:

- ¿Hasta qué parte del log debo leer?
- ¿Qué variables deben deshacer sus cambios y cu´al es el valor con el que quedarán?
- ¿Qué variables (de las que aparecen en el log) no son cambiadas en el proceso?

Log Undo
<START T1>
<START T2>
<T1, a, 4>
<T2, b, 5>
<T2, c, 10>
<COMMIT T1>
<START CKPT (T2)>
<START T3>
<START T4>
<T3, a, 10>
<T2, b, 7>
<T4, d, 5>
<COMMIT T2>
<END CKPT>
<START T5>
<COMMIT T3>
<T5, e, -3>

Undo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Undo”. Suponiendo que la política de recovery es la de Undo Logging, indique:

- ¿Hasta qué parte del log debo leer?
- ¿Qué variables deben deshacer sus cambios y cu´al es el valor con el que quedarán?
- ¿Qué variables (de las que aparecen en el log) no son cambiadas en el proceso?

Log Undo
<START T1>
<START T2>
<T1, a, 4>
<T2, b, 5>
<T2, c, 10>
<COMMIT T1>
<START CKPT (T2)>
<START T3>
<START T4>
<T3, a, 10>
<T2, b, 7>
<T4, d, 5>
<COMMIT T2>
<END CKPT>
<START T5>
<COMMIT T3>
<T5, e, -3>

Undo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Undo”. Suponiendo que la política de recovery es la de Undo Logging, indique:

- ¿Hasta qué parte del log debo leer?

Dado que existe un END CKPT, se debe leer hasta el START CKPT.

Log Undo
<START T1>
<START T2>
<T1, a, 4>
<T2, b, 5>
<T2, c, 10>
<COMMIT T1>
<START CKPT (T2)>
<START T3>
<START T4>
<T3, a, 10>
<T2, b, 7>
<T4, d, 5>
<COMMIT T2>
<END CKPT>
<START T5>
<COMMIT T3>
<T5, e, -3>

Undo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Undo”. Suponiendo que la política de recovery es la de Undo Logging, indique:

- ¿Qué variables deben deshacer sus cambios y cuál es el valor con el que quedarán?

T5 no está marcada con commit, por lo que por la línea indica que debemos hacer que e vuelva a ser -3. Por la misma razón, la línea nos indica que d debe volver a ser 5.

Log Undo
<START T1>
<START T2>
<T1, a, 4>
<T2, b, 5>
<T2, c, 10>
<COMMIT T1>
<START CKPT (T2)>
<START T3>
<START T4>
<T3, a, 10>
<T2, b, 7>
<T4, d, 5>
<COMMIT T2>
<END CKPT>
<START T5>
<COMMIT T3>
<T5, e, -3>

Undo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Undo”. Suponiendo que la política de recovery es la de Undo Logging, indique:

- ¿Qué variables (de las que aparecen en el log) no son cambiadas en el proceso?

No se debe hacer undo de ninguna otra transacción, por lo que las variables **a, b y c** no son tocadas.

Log Undo
<START T1>
<START T2>
<T1, a, 4>
<T2, b, 5>
<T2, c, 10>
<COMMIT T1>
<START CKPT (T2)>
<START T3>
<START T4>
<T3, a, 10>
<T2, b, 7>
<T4, d, 5>
<COMMIT T2>
<END CKPT>
<START T5>
<COMMIT T3>
<T5, e, -3>

Undo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Undo”. Suponiendo que la política de recovery es la de Undo Logging, indique:

- ¿Qué variables deben deshacer sus cambios y cual es el valor con el que quedarían?

Log Undo
<START T1>
<START T2>
<T1, a, 4>
<T2, b, 5>
<T2, c, 10>
<COMMIT T1>
<START CKPT (T2)>
<START T3>
<START T4>
<T3, a, 10>
<T2, b, 7>
<T4, d, 5>
<COMMIT T2>
<END CKPT>
<START T5>
<COMMIT T3>
<T5, e, -3>

Undo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Undo”. Suponiendo que la política de recovery es la de Undo Logging, indique:

- ¿Qué variables no son cambiadas en el proceso?

Log Undo
<START T1>
<START T2>
<T1, a, 4>
<T2, b, 5>
<T2, c, 10>
<COMMIT T1>
<START CKPT (T2)>
<START T3>
<START T4>
<T3, a, 10>
<T2, b, 7>
<T4, d, 5>
<COMMIT T2>
<END CKPT>
<START T5>
<COMMIT T3>
<T5, e, -3>

Redo Loggin

Regla 1:

Antes de modificar cualquier elemento X en el disco, es necesario que todos los logs esten almacenados en disco:

1. Escribir el log <**T,** X, v>
2. Escribir <COMMIT **T**>
3. Escribir los datos en disco.
4. Escribir <T, End> en log.

Redo Loggin

Recovery con checkpoints:

Procesar el log desde el final al inicio:

- Si hay un <END CKPT>, retroceder hasta el respectivo <START CKPT> y comenzar a hacer redo desde la transacción más antigua sin END.
- No hacer redo de las transacciones con COMMIT antes del <START CKPT>.

Redo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Redo”. Suponiendo que la política de recovery es la de Redo Logging, indique:

Log Redo
<START T1>
<T1, a, 1>
<COMMIT T1>
<START T2>
<T2, b, 2>
<T2, c, 3>
<COMMIT T2>
<START T3> <T3, a, 10>
<START CKPT (T3)>
<END T1>
<END T2>
<T3, d, 23>
<START T4>
<END CKPT>
<COMMIT T3>
<T4, e, 11>

- ¿Desde qué parte del log debo comenzar el proceso de redo?
- ¿Qué variables deben rehacer sus cambios y cuál es el valor con el que quedarán?
- Qué variables (de las que aparecen en el log) no son cambiadas en el proceso?
- Si no hubiésemos encontrado la línea, ¿desde qué parte del log deberíaa comenzar el proceso de redo?

Redo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Redo”. Suponiendo que la política de recovery es la de Redo Logging, indique:

- ¿Desde qué parte del log debo comenzar el proceso de redo?

Log Redo
<START T1>
<T1, a, 1>
<COMMIT T1>
<START T2>
<T2, b, 2>
<T2, c, 3>
<COMMIT T2>
<START T3> <T3, a, 10>
<START CKPT (T3)>
<END T1>
<END T2>
<T3, d, 23>
<START T4>
<END CKPT>
<COMMIT T3>
<T4, e, 11>

Dado que existe un END CKPT:

Vemos que la transacción más antigua del checkpoint es T3 (<START CKPT (T3)>), por ende leemos hasta **<START T3>**.

Redo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Redo”. Suponiendo que la política de recovery es la de Redo Logging, indique:

- ¿Qué variables deben rehacer sus cambios y cuál es el valor con el que quedarán?

Log Redo
<START T1>
<T1, a, 1>
<COMMIT T1>
<START T2>
<T2, b, 2>
<T2, c, 3>
<COMMIT T2>
<START T3> <T3, a, 10>
<START CKPT (T3)>
<END T1>
<END T2>
<T3, d, 23>
<START T4>
<END CKPT>
<COMMIT T3>
<T4, e, 11>

Dado que existe un END CKPT, tenemos la certeza de que T1 y T2 estan están guardadas en disco. Por ende rehacemos las transacciones marcadas con COMMIT (T3).

Redo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Redo”. Suponiendo que la política de recovery es la de Redo Logging, indique:

- ¿Qué variables deben rehacer sus cambios y cuál es el valor con el que quedarán?

Log Redo
<START T1>
<T1, a, 1>
<COMMIT T1>
<START T2>
<T2, b, 2>
<T2, c, 3>
<COMMIT T2>
<START T3> <T3, a, 10>
<START CKPT (T3)>
<END T1>
<END T2>
<T3, d, 23>
<START T4>
<END CKPT>
<COMMIT T3>
<T4, e, 11>

<T3, a, 10>

<T3, d, 23>

Rehacemos:

- **a** al **valor 10**
- **d** al **valor 23**.

Redo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Redo”. Suponiendo que la política de recovery es la de Redo Logging, indique:

- ¿Qué variables (de las que aparecen en el log) no son cambiadas en el proceso ?

Log Redo
<START T1>
<T1, a, 1>
<COMMIT T1>
<START T2>
<T2, b, 2>
<T2, c, 3>
<COMMIT T2>
<START T3> <T3, a, 10>
<START CKPT (T3)>
<END T1>
<END T2>
<T3, d, 23>
<START T4>
<END CKPT>
<COMMIT T3>
<T4, e, 11>

Dado que T4 no esta marcado con COMMIT, **e** no se debe tocar y se debe hacer ABORT.

a, b y c tampoco son cambiadas durante el proceso de recovery.

Redo Loggin

Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el log file que se muestra a continuación, en la tabla ”Log Redo”. Suponiendo que la política de recovery es la de Redo Logging, indique:

- ¿Si no hubíesemos encontrado la línea <END CKPT>, ¿desde qué parte del log debería comenzar el proceso de redo?

Log Redo
<START T1>
<T1, a, 1>
<COMMIT T1>
<START T2>
<T2, b, 2>
<T2, c, 3>
<COMMIT T2>
<START T3> <T3, a, 10>
<START CKPT (T3)>
<END T1>
<END T2>
<T3, d, 23>
<START T4>
~~<END CKPT>~~
<COMMIT T3>
<T4, e, 11>

Cuando no hay END CKPT se debe leer el log entero.

Gracias!