Transactions (Partie II)

G. Raschia

dernière mise à jour : le 9 décembre 2022

1 2PL

Soit le plan d'exécution P des trois transactions T_1, T_2 et T_3 :

$$P = r_1(a); r_2(c); r_2(b); w_1(a); r_3(a); r_1(b); w_2(c); w_2(b); c_2; r_3(b); r_3(c); c_3; w_1(b); c_1$$

1. Énumérer dans P les paires d'actions conflictuelles pouvant produire chacune des anomalies suivantes :
   (a) les « lectures sales » (*dirty reads*)
   (b) les « lectures uniques » (*unrepeatable reads*)
   (c) les « modifications perdues » (*lost updates*)

   Solution :
   (a) $(w_1(a), r_3(a))$ $(w_2(c), r_3(c))$ $(w_2(b), r_3(b))$
   (b) $(r_1(b), w_2(b))$ $(r_2(b), w_1(b))$ $(r_3(b), w_1(b))$. Les 3 paires $(r_2(c), w_2(c))$, $(r_1(b), w_1(b))$ et $(r_2(b), w_2(b))$ portent sur la même transaction. Elles ne sont pas conflictuelles et **ne peuvent en aucun cas** être permutées. En effet, la séquence d'actions qui compose une transaction est, par définition, immuable.
   (c) $(w_2(b), w_1(b))$

2. Construire le graphe de sérialisation $\mathcal{G}(P)$ du plan P.

   Solution :
   à faire, à partir des dépendances $T_i \rightarrow T_j$ obtenues si (f_i, f_j) est une paire d'actions conflictuelles (cf. question 1).

3. P est-il sérialisable par conflit ? Justifier.

   Solution :
   non. cycles (T_1, T_2), (T_1, T_3) et (T_1, T_2, T_3) !

4. On s'intéresse au protocole de verrouillage à deux phases (2PL). Les modes exclusif $x_i(.)$ et partagé $s_i(.)$ sont disponibles, ainsi que l'opération de libération $\ell_i(.)$ et la promotion de verrou s en verrou x, notée $x_i^*(.)$. P est-il 2PL ?

   Solution :
   Non. Tous les plans 2PL sont nécessairement sérialisables par conflit. L'inverse n'est pas vrai.

On rappelle d'abord que la promotion de verrou ne viole pas 2PL, autrement dit, elle n'est pas considérée comme la libération d'un verrou s puis la prise d'un verrou x. En outre, un verrou x^* est strictement identique à un verrou x.

Après l'écriture $w_1(a)$, les verrous $x_1^*(a)$, $s_2(c)$, $s_2(b)$ sont posés. T_3 a besoin d'un verrou s sur a qu'elle ne peut obtenir tant que T_1 possède $x_1^*(a)$. Or, de deux choses l'une : soit le système reçoit les opérations en flux et il est alors impossible de prévoir le moment où T_1 poourra commencer à libérer ses verrous avant le commit (car on ne sait jamais si T_1 va demander de nouvelles opérations). Dans ce cas, T_1 ne peut libérer $x_1^*(a)$ et P n'est pas 2PL (mise en attente de T_3). Soit, le contrôleur a déjà reçu toute l'histoire et dans ce cas, on peut tenter de raisonner un peu plus : pour que T_1 libère le verrou $x_1^*(a)$, il lui faut préalablement obtenir un verrou x sur b en prévision d'une lecture-écriture à venir, et en conformité avec 2PL. Pour cela, il est nécessaire que T_2 libère $s_2(b)$, et par conséquent que T_2 obtienne préalablement $x_2(c)$—pas de problème—et $x_2^*(b)$ pour l'écriture $w_2(b)$ ultérieure. En résumé, T_1 requiert x sur b qui ne peut lui être fourni que si T_2 obtient x sur b ! L'histoire P n'est donc pas 2PL.

5. Rejouer l'histoire P selon 2PL. Si une action ne peut accéder à une ressource, la transaction correspondante est mise en attente et l'action suivante du plan est considérée. Reporter les problèmes éventuels.

Solution :

$$\begin{aligned} P = & s_1(a); r_1(a); s_2(c); r_2(c); s_2(b); r_2(b); x_1^*(a); w_1(a); [s_3(a); r_3(a)]; \\ & s_1(b); r_1(b); x_2^*(c); w_2(c); [x_2^*(b); w_2(b)]; [x_1^*(b); w_1(b)] \end{aligned}$$

Interblocage !

Un verrou de mise-à-jour $u_i(.)$ est une forme spécifique de verrou partagé, qui, une fois posé, interdit toute pose de verrou supplémentaire, partagé ou exclusif. En revanche, il n'est pas impossible que la ressource soit verrouillée en lecture avant la pose du verrou de mise-à-jour. À quoi sert un verrou de mise-à-jour ? Il est utile en combinaison avec la promotion de verrou $x_i^*(.)$. En effet, chaque fois qu'une transaction lit un objet dans la perspective de l'écrire, le programmateur pose un verrou de mise-jour plutôt qu'un verrou partagé, réalise la lecture, puis il promeut ce verrou de mise-à-jour en verrou exclusif au moment de l'écriture.

6. Rejouer l'histoire P selon 2PL, en considérant que l'on dispose maintenant de verrous de mise-à-jour.

Solution :

T_1 pose des verrous de mise-à-jour qui mettent les autres transactions en attente. Ensuite tout se passe bien.

	T_1	T_2	T_3
01	$u_1(a); r_1(a)$		
02		$u_2(c); r_2(c)$	
03	$u_1(b); r_1(b)$		
04	$x_1^*(a); w_1(a)$		
-			$[s_3(a)]$
-		$[u_2(b)]$	
05	$x_1^*(b); w_1(b)$		
06	$\ell_1(a); \ell_1(b)$		
07			$s_3(a); r_3(a)$
08		$u_2(b); r_2(b)$	
09		$x_2^*(c); w_2(c)$	
10		$x_2^*(b); w_2(b)$	
11		$\ell_2(b); \ell_2(c)$	
12			$s_3(b); r_3(b)$
13			$s_3(c); r_3(c)$
14			$\ell_3(b); \ell_3(c)$

2 Journalisation

Soient les 12 enregistrements d'un journal des images Avant/Après :

01	$\langle \text{START } T_1 \rangle$
02	$\langle T_1, a, 4, 5 \rangle$
03	$\langle \text{START } T_2 \rangle$
04	$\langle \text{COMMIT } T_1 \rangle$
05	$\langle T_2, b, 9, 10 \rangle$
06	$\langle \text{START CKPT } (T_2) \rangle$
07	$\langle \text{START } T_3 \rangle$
08	$\langle T_3, a, 5, 17 \rangle$
09	$\langle T_2, a, 17, 4 \rangle$
10	$\langle \text{END CKPT} \rangle$
11	$\langle \text{COMMIT } T_2 \rangle$
12	$\langle \text{COMMIT } T_3 \rangle$

Un point de reprise (`CKPT`) a une durée marquée par les instants de début (`START CKPT`) et de fin (`END CKPT`). Toutes les transactions actives au début du point de reprise sont indiquées dans l'enregistrement `START CKPT`. Au point de reprise, l'écriture des pages sales (opération *flush*) est garantie pour toutes les actions antérieures à l'enregistrement `START CKPT`. Cependant, tant que l'enregistrement de fin de point de reprise (`END CKPT`) n'est pas écrit dans le journal, le point de reprise en cours n'offre *aucune* garantie.

1. Décrire la procédure de reprise lorsqu'une panne survient :
   (a) après l'enregistrement 12 ;
   (b) après l'enregistrement 11 ;
   (c) après l'enregistrement 10 ;
   (d) après l'enregistrement 09.

Solution :

(a) Un point de reprise valide est signalé par un enregistrement de début et de fin. Il garantit que toutes les modifications marquées *avant* le début du point de reprise ont été inscrites dans la base de données.

T_1 a été commise avant le début du point de reprise, donc il est inutile de s'en préoccuper. T_2 et T_3 sont validées avant la panne. Depuis l'enregistrement 06 (début du point de reprise), il faut rejouer T_3, donc écrire $a = 17$. De même il faut rejouer T_2. Les actions préalables à l'enregistrement 06 ont été prises en compte (avec certitude) par la BD. Donc on ne rejoue que $a = 4$. Attention à bien respecter l'ordre chronologique des enregistrements de journal : rejouer (T_3, a) d'abord, (T_2, a) ensuite. On voit bien sur cet exemple qu'inverser les écritures conduit à un état incohérent de la base car la dernière écriture de a doit être celle de T_2 ($a = 4$).

(b) Dans ce cas, T_3 n'est pas validée au moment de la panne. Il faut défaire T_3 ($a = 5$) dans une lecture antéchronologique du journal et ajouter un enregisterment $\langle \text{ABORT } T_3 \rangle$, puis rejouer partiellement T_2 ($a = 4$). Là encore, l'ordre des actions a son importance pour la cohérence de la base de données (dernière écriture de a).

(c) Ni T_2 ni T_3 ne sont validées. Il faut donc les déjouer toutes les deux, dans l'ordre antéchronologique du journal, soit $a = 17$ (T_3), $a = 5$ (T_2) et $b = 9$ (T_2). Ne pas oublier de remonter le journal jusqu'au début de la transaction T_2, donc au-delà

du début du point de reprise car dans ce cas précis, il est certain que l'écriture sur b par T_2 a été reportée dans la base de données. En outre, il faut ajouter les enregistrements $\langle \text{ABORT } T_2 \rangle$ et $\langle \text{ABORT } T_3 \rangle$. L'ordre de ces 2 écritures dans le journal n'a pas d'importance dès lors que les transactions avec une marque ABORT sont systématiquement déjouées lors d'une reprise. Dans le cas contraire, une panne lors de la reprise pourrait corrompre la base de données !

(d) L'extrait de journal n'est pas suffisant pour décider. Le dernier point de reprise valide n'est pas donné. Il n'y a aucune garantie quant au fait que toutes les pages sales aient été écrites sur support stable au point de reprise démarré en 06.

Dans l'hypothèse—peu vraisemblable en pratique—où le journal est complet, il faut alors déjouer T_3, déjouer T_2, puis rejouer T_1.