Gestion de la concurrence d’accès

Alexandre Clenet - Florian Tran
Année 3 : Groupe 1-2

Exercice 1

Indiquer pour chacune de ces questions le résultat de l’exécution des transactions T1 et T2 (sérialisable, perte de mise à jour, etc.) .

a)
T1:find S T1:find C1 T1:upd S T1:upd C1 T2:find S T2:find C2 T2:upd S T2:upd C2
serialisable

b)
T1:find S
T1:find C1
T2:find S
T2:find C2
T2:upd S
T2:upd C2
T1:upd S
T1:upd C1
perte de mise à jour

c)
T1:find S
T1:find C1
T1:upd S
T2:find S
T2:find C2
T2:upd S
T1:upd C1
T2:upd C2
serialisable

d)
T1:find S
T1:find C1
T1:upd S
T2:find S
T2:find C2
T2:upd S
T2:upd C2
T2:commit
T1:upd

C1
T1:Rollback
problème de lecture sale

e)
T1:find S
T2:find S
T1:upd S
T2:find C2
T2:find S
T1:Commit
T2:upd C2
T2:Commit
problème de lecture non reproductible

Exercice 2

Nous considérons une séquence d'opérations impliquant des transactions T1,T2,T3,T4 où A et B sont deux granules de la base de données. On suppose que T1,T2,T3 et T4 suivent respectivement les protocoles PSE, PS, PX et PUE (vus en cours). Compléter le tableau suivant en indiquant à chaque instant t l'état de la pile et le graphe d'attente entre transactions.

TP (PSE)	T2(PS)	T3 (PX)	T4 (PUE)	Temps	pile de verrou	attentes
D				t1		
	D			t2		
			D	t3		
Find A				t4	T1,SE,A	
			Find B	t5	T4,UE,B	
	Find B			t6	T2,S,B	
		D		t7		
	Find A			t8	T2,S,A	
			Upd B	t9	T4,X,B	T4(B) -> T2(B)
		Find B		t10	T3,X,B	T3(B) -> T4(B) -> T2(B)
Find B				t11	T1,SE,B	T3(B) -> T4(B) -> T2(B)
	Udp B			t12	T2,X,B	T3(B) -> T4(B)
			Commit	t13		
	Commit			t14		
		Find A		t15	T3,X,A	
Commit				t16		

TP (PSE)	T2(PS)	T3 (PX)	T4 (PUE)	Temps	pile de verrou	attentes
		Commit		t17		

Exercice 3

On considère le tableau de l'exercice précédent. Compléter ce tableau en adoptant une approche préventive (Wait-Die / Wound-Wait). Comparer le résultat de ces deux approches.

Wait - Die

TP (PSE)	T2(PS)	T3 (PX)	T4 (PUE)	Temps	pile de verrou	attentes
D				t1		
	D			t2		
			D	t3		
Find A				t4	T1,SE,A	
			Find B	t5	T4,UE,B	
	Find B			t6	T2,S,B	
		D		t7		
	Find A			t8	T2,S,A	
			Upd B	t9	T4,X,B	T4 annulé car plus jeune que T2
		Find B		t10	T3,X,B	T3 annulé car plus jeune que T2
Find B				t11	T1,SE,B	T1/T2(B)
	Udp B			t12	T2,X,B	T2 annulé car plus jeune que T1
			Commit	t13		
	Commit			t14		
		Find A		t15	T3,X,A	
Commit				t16		
		Commit		t17		

Exercice 4

Vous ouvrez deux fenêtres SQLDeveloppper pour avoir deux transactions concurrentes. Vous appelez vos connexions fenetre1et fenetre2. Vous faites ce qui est indiqué ci-dessous. Vous expliquez ce qui se passe et vous justifiez vos réponses. Les questions sont indépendantes : les modifications apportées à la base dans une question ne sont pas prises en compte dans les autres questions (sauf indication).

1:
Avec l'insertion la fenêtre 1 n'a pas accès a la modification tant que la fenêtre 2 n'a pas commit

2:
Meme chose la suppression est effective pour la fenêtre 1 seulement apres un commit de la fenêtre 2

3:
Pour la fenêtre avec la transaction en read only, la lecture doit être reproductible donc meme avec une modification et commit de la fenêtre 2, la fenêtre 1 garde la meme lecture

4:
L'update de la fenêtre 2 n'est pas effective car la fenêtre 1 n'a pas encore commit la création de l'employé qui doit être mis a jour donc l'update est vide.

5:
La fenêtre 1 delete l'employé qui le rend inaccessible pour l'update de celui ci par la fenêtre 2 (ca tourne dans le vide). Au moment du rollback, l'update est directement faite car la fenêtre 2 continuait de chercher l'employé à update.

6:
La suppression de l'employé par la fenêtre 1 empêche directement la création d'une nouvelle affectation de celui-ci dans Travail car la clé est introuvable.

7:
La fenêtre 1 rollback la création de Dupond donc l'update de la fenêtre 2 sur Dupond n'est pas possible

8:
La fenêtre 1 select l'employé 100 pour une mise à jour. C'est ensuite la fenêtre 2 qui fait de même mais elle est mise en attente car la fenêtre 1 a deja engagé ce protocole qui n'est pas compatible (U). La fenêtre 1 fait sa modification commit puis c'est au tour de la fenêtre 2 de faire sa modification.

9:
Les 2 fenêtres select respectivement l'employé 100 et 101 pour une mise à jour comme précédemment n'étant pas sur le meme employé aucune mise en attente n'est requise les modifications sont effectué en parrallèle sans problèmes

10:
La fenêtre 1 verouille la table employé en mode partage donc la fenêtre 2 est en file d'attente pour son insertion. Une fois le commit fait, la fenêtre 2 fait son insertion.

11:
La fenêtre 1 verouille la table employé en mode partage donc la fenêtre 2 est en file d'attente pour son insertion. Une fois le commit fait, la fenêtre 2 fait son insertion.

12:
La fenêtre 2 verouille les tables employé et travail en mode partage donc la fenêtre 1 est en file d'attente pour son insertion. Une fois l'insertion et le commit fait, la fenêtre 1 fait son insertion.

13:
La fenêtre 1 verouille la table employé en mode partage ainsi que la fenêtre 2 mais pour une seule ligne la fenêtre 2 peux donc faire sa modification sur sa ligne voulu et la fenêtre 1 fait aussi son update en parrallèle.

14:

Les fenêtre 1 et 2 verouillent la table employé en mode partage pour une seule ligne. La fenêtre 2 peux donc faire sa modification sur sa ligne voulu et la fenêtre 1 fait aussi son update en parrallèle.

15:

La fenêtre 1 verrouille la table employé en mode exclusif donc la fenêtre 2 doit attendre le commit de la fenêtre 1 avant de pouvoir mettre son verrou

16:

Les 2 fenêtres selectionnent le même employé pour une mise à jour. Priorité à la fenêtre 2 et donc la fenêtre 1 qui doit attendre le commit

17:

La fenêtre 1 verrouille la table employé en mode exclusif pour une ligne et la fenêtre 2 verrouille la table employé en mode partagé pour une ligne. Les 2 premiers update ne pose aucun problème car c'est sur 2 employés distincts. Pour le 2eme update de la fenêtre 1, il est en attente du commit car c'est l'employé modifié par la fenêtre 2 (Row Exclusive et Row Shared pas compatible)