SGBD : BASES DE DONNÉES AVANCÉES [M3106C]

TD $N^0$3 - ACCÈS CONCURRENTS

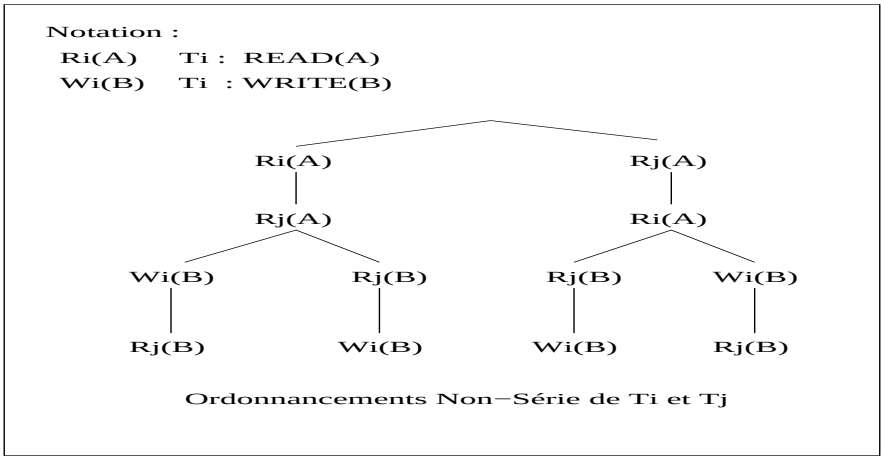
Objectifs

- Ordonnancements sérialisables
- Méthode d'accès par verrouillage à deux phases
- Méthodes d'accès par estampillage à l'initialisation

Corrigés

Exercice I :

Question 1.1.

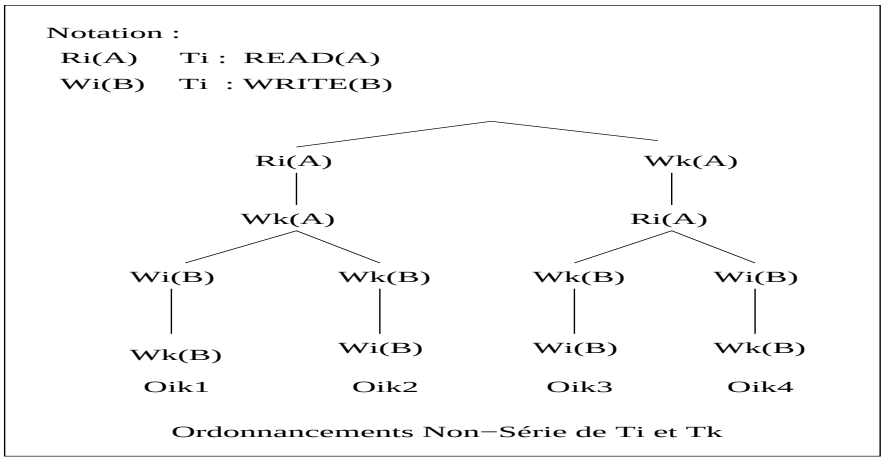


Il y a 4 ordonnancements : tous sont sérialisables.

Question 1.2.

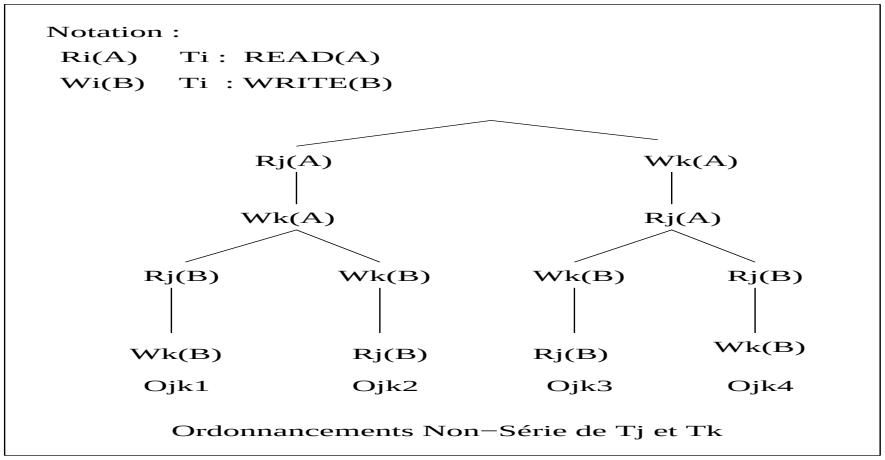
Date: 9 septembre 2014.

Hocine ABIR - IUT Villetaneuse .



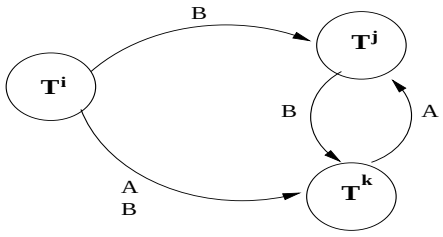
Il y a 4 ordonnancements : Seuls Oik1 et Oik3 sont sérialisables.

Question 1.3.



Il y a 4 ordonnancements : Seuls Ojk1 et Ojk3 sont sérialisables.

Question 1.4.



Question 1.5.

On note [] les verrous acquis et {} les verrous non-acquis

1)Premier COMMIT : Ti a terminé et a tous ses verrous actifs.

T_i	T_j	T_k	A	B
READ(A)			$[T_i$,R]	
		WRITE(A)	$[T_i$,R] $\{T_k$,W$\}$	
	READ(A)		$[T_i$,R] $[T_j$,R] $\{T_k$,W$\}$	
WRITE(B)			$[T_i$,R] $[T_j$,R] $\{T_k$,W$\}$	$[T_i$,W]
commit			$[T_j$,R] $\{T_k$,W$\}$	
	READ(B)		$[T_j$,R] $\{T_k$,W$\}$	$[T_j$,R]
	commit		$[T_k$,W]	
		WRITE(A)	$[T_k$,W]	
		WRITE(B)	$[T_k$,W]	$[T_k$,W]
		commit		

2)on enlève tous les verrous actifs de Ti

3)deuxième COMMIT : Tj a terminé et a tous ses verrous actifs.

4)on enlève tous les verrous actifs de Tj

6)on enlève tous les verrous actifs de Tk

Il y a un 2ème exemple page 5 questions 3.3

5)troisième COMMIT : Tk a terminé et a tous ses verrous actifs

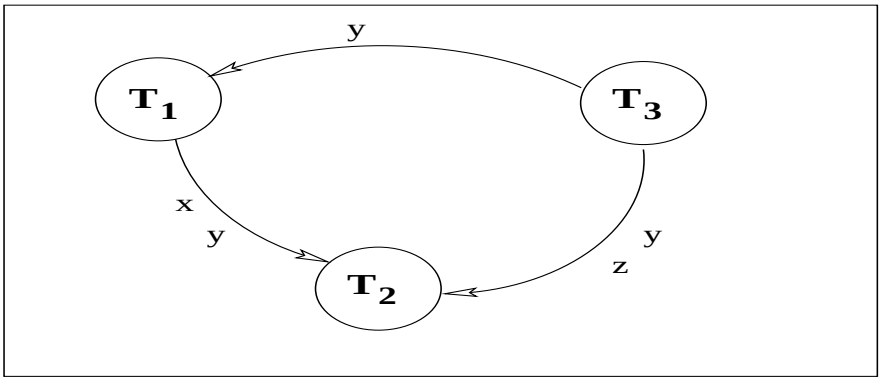
L'ordonnancement série est : T_i , T_j, T_k

Question 1.6.

l'accès aux ressources A et B se fait dans le même ordre pour les 3 transactions.

Exercice II :

Question 2.1.



Question 2.2.

- S est sérializable : car son graphe de précédence ne contient pas de cycle.
- S est équivalent à l'ordonnancement série $< T_3, T_1, T_2 >$

Question 2.3.

VOIR PAGE 5
avant de lire ici

Temps	T_1	T_2	X	Y	Z
1	READ(X)		[T1,R]		
2	WRITE(X)		[T1,R],[T1,W]		
3		READ(Z)	[T1,R],[T1,W]		[T2,R]
4		READ(Y)	[T1,R],[T1,W]	[T2,R]	[T2,R]
5		WRITE(Y)	[T1,R],[T1,W]	[T2,W],[T2,R]	[T2,R]
6	READ(Y)		[T1,R],[T1,W]	[T2,W],[T2,R],[T1,R]	[T2,R]
7	WRITE(Y)		[T1,R],[T1,W]	[T2,W],[T2,R],[T1,R],{T1,W}	[T2,R]
8		READ(X)			
9		WRITE(X)			

T1 a terminé mais n'a pas tous ses verrous (en rouge) actifs, on ne peut pas faire de COMMIT ici

idem ici en continuant, T2 termine mais n'aura pas le verrou {T2,W} actif donc ne peut pas faire de COMMIT : blocage mutuel.
Dans X on aura [T1,R],[T1,W], [T2,R], {T2,W}
Dans Y on aura [T2,W],[T2,R],[T1,R],{T1,W}
on a donc T2 qui attend T1 pour écrire sur X, T1 qui attend T2 pour écrire sur Y

Exercice III :

Question 3.1.

(1) Solution 1
Dans l'ordonnancement série $< T_{i1}, T_{i2} >$, l'opération de (2) de T_{i1} et l'opération (3) T_{i2} (voir tableau ci-dessous) ne sont pas permutables donc aucun ordonncement n'est sérialisable.

Opération	T_{i1}	T_{i2}
1	READ(Y)	
2	WRITE(Y)	
3		READ(Y)
4		WRITE(Y)

idem pour $< T_{i2}, T_{i1} >$

(2) Solution 2
Enumérer tous les ordonnancements non-séries et vérifier que chacun d'eux comporte un cycle dans le graphe de précédence :

#	Ordonnancement	Cycle
1	$R_{i1}(Y)$, $R_{i2}(Y)$, $W_{i1}(Y)$, $W_{i2}(Y)$	$R_{i1}(Y) < W_{i2}(Y)$ et $R_{i2}(Y) < W_{i1}(Y)$
2	$R_{i1}(Y)$, $R_{i2}(Y)$, $W_{i2}(Y)$, $W_{i1}(Y)$	$R_{i1}(Y) < W_{i2}(Y)$ et $R_{i2}(Y) < W_{i1}(Y)$
3	$R_{i2}(Y)$, $R_{i1}(Y)$, $W_{i2}(Y)$, $W_{i1}(Y)$	$R_{i2}(Y) < W_{i1}(Y)$ et $R_{i1}(Y) < W_{i2}(Y)$
4	$R_{i2}(Y)$, $R_{i1}(Y)$, $W_{i1}(Y)$, $W_{i2}(Y)$	$R_{i2}(Y) < W_{i2}(Y)$ et $R_{i1}(Y) < W_{i2}(Y)$

Question 3.2.

comme on veut des ordonnancements sérialisables, on ne peut pas échanger les opérations non-commutatives WRITE et READ

De chaque ordonnancement série de T_{j1} et T_{j2}, on peut construire deux ordonnancements non-séries sérialisables équivalents.

#	Ord Série	Ordonnancements Non-Série Sérialisables
1	$< T_{j1}, T_{j2} >$	$w_{j1}(X)$, $R_{j1}(Y)$, $w_{j2}(X)$, $W_{j1}(Y)$,$R_{j2}(Y)$, $W_{j2}(Y)$
2		$w_{j1}(X)$, $w_{j2}(X)$, $R_{j1}(Y)$, $W_{j1}(Y)$, $R_{j2}(Y)$, $W_{j2}(Y)$
3	$< T_{j2}, T_{j1} >$	$w_{j2}(X)$, $R_{j2}(Y)$, $w_{j1}(X)$, $W_{j2}(Y)$, $R_{j1}(Y)$, $W_{j1}(Y)$
4		$w_{j2}(X)$, $w_{j1}(X)$, $R_{j2}(Y)$, $W_{j2}(Y)$, $R_{j1}(Y)$, $W_{j1}(Y)$

Autre Solution :
Enumérer tous les ordonnancements (Il y a au total 18) et vérifier pour chacun d'eux la sérialisibilité.

Question 3.3.

Premier COMMIT : Tj1 a terminé et a tous ses verrous (en jaune) actifs, on les enlève donc tous

Deuxième COMMIT : Tj2 a terminé et a tous ses verrous (en bleu) actifs, on les enlève donc tous

ce verrou n'est pas actif car il a d'abord été ouvert par Tj1

ce verrou est actif car Y n'est ouvert que par Tj1

T_{j1}	T_{j2}	X	Y
WRITE(X)		[j1,W]	
	WRITE(X)	[j1,W] , {j2,W}	
READ(Y)		[j1,W] , {j2,W}	[j1,R]
WRITE(Y)		[j1,W] , {j2,W}	[j1,R] , [j1,W]
COMMIT		[j2,W]	
	WRITE(X)	[j2,W]	
	READ(Y)	[j2,W]	[j2,R]
	WRITE(Y)	[j2,W]	[j2,R] , [j2,W]
	COMMIT		

Question 3.4.

- La transaction T_j débute avec un WRITE(X),
- L'occurrence de T_j qui débute obtient un verrou WRITE (exclusif) sur X et bloque la deuxième occurrence jusqu'à son COMMIT.

Question 3.5.

- tous les ordonnancements non série entrainent un blocage mutuel
- La transaction T_k débute avec un READ(Y),
- Chaque occurrence de T_k obtient un verrou READ (partagé) sur Y et les deux occurrences s'attendent mutuellement.
- Exemple :

ce verrou ne peut pas être actif car c'est une autre transaction (Tk2) qui a ouvert la ressource Y avec READ(Y)

ce verrou ne peut pas être actif car c'est une autre transaction (Tk1) qui a ouvert la ressource Y avec READ(Y)

T_{k1}	T_{k2}	X	Y
READ(Y)			[j1,R]
	READ(Y)		[j1,R] , [j2,R]
WRITE(Y)			[j1,R] , [j2,R] , {j1,W}
	WRITE(Y)		[j1,R] , [j2,R] , {j1,W} , {j2,W}
...			blocage mutuel
READ(X)		[j1,R]	[j1,R], [j2,R], {j1,W}, {j2,W}
	READ(X)	[j1,R][j2,R]	[j1,R], [j2,R], {j1,W}, {j2,W}

Tk1 a terminé mais n'a pas tous ses verrous (en rouge) actifs, on ne peut pas faire de COMMIT ici

Tk2 a terminé mais n'a pas tous ses verrous (en orange) actifs, on ne peut pas faire de COMMIT ici

du coup les deux transactions s'attendent, blocage mutuel : Tk2 attend Tk1 pour écrire sur Y, et Tk1 attend Tk2 pour écrire sur Y