

Exercise 04

- L'ordonnancement O: R1(A) R2(A) R2(B) R3(A) R3(B) C3 W1(A) R1(B) W1(B) C1 W2(A) W2(B) C2

[illegible]

Exercice 04

4. Donner le scénario d'exécution de cet ordonnancement en utilisant la technique d'estampillage à deux estampilles.

L'ordonnancement **O: R1(A) R2(A) R2(B) R3(A) R3(B) C3 W1(A) R1(B) W1(B) C1 W2(A) W2(B) C2**

Transaction	Action	EL(A)	EE(A)	EL(B)	EE(B)	Commentaire
		3	0	3	0	
Relancer T1 → T4						
T4	R4(A)	4	0	3	0	E(T4)=4; EL(A)=3 ;EE(A)=0; E(T4) >= EE(A) → R4(A) est acceptée et EL(A)=Max(EL(A), E(T4))=4
T4	W4(A)	4	4	3	0	E(T4)=1; EL(A)=4; EE(A)=0; E(T4) >= EE(A) et E(T4) >= EE(A) → W4(A) est acceptée et EE(A)=E(T4)=4
T4	R4(B)	4	4	4	0	E(T4)=1; EL(B)=3 ;EE(B)=0; E(T4) >= EE(B) → R4(B) est acceptée et EL(B)=Max(EL(B), E(T4))=4
T4	W4(B)	4	4	4	4	E(T4)=1; EL(B)=4; EE(B)=0; E(T4) >= EE(B) et E(T4) >= EE(B) → W4(B) est acceptée et EE(B)=E(T4)=4
Relancer T2 → T5						
T5	R5(A)	5	4	4	4	
T5	R5(B)	5	4	5	4	
T5	W5(A)	5	5	5	4	
T5	W5(B)	5	5	5	5	

Exercice 04

L'ordonnancement O: R1(A) R2(A) R2(B) R3(A) R3(B) C3 W1(A)
R1(B) W1(B) C1 W2(A) W2(B) C2

5. Si la stratégie wait-die est utilisée pour gérer les demandes de verrouillage. Quelle est l'exécution finale obtenue ?

Algorithme wait-die

Soient 2 transactions T_i , T_j , un granule r utilisé par T_i et demandé par T_j , et soient les estampilles $E(T_i)$ et $E(T_j)$ associées aux deux transactions.

Algorithme: Wait- Die

Si $E(T_j) < E(T_i)$ alors bloquer T_j /*Wait*/

sinon annuler T_j /*Die*/

/* T_j redémarrée ultérieurement avec la même estampille*/

Fsi

- Dans la stratégie de prévention de dead-lock «wait-die », une transaction plus ancienne est autorisée à attendre une transaction plus jeune, tandis qu'une transaction plus jeune qui demande un granule détenu par une transaction plus ancienne est annulée et redémarrée ultérieurement avec la même estampille.

Exercice 04

L'ordonnancement O: R1(A) R2(A) R2(B) R3(A) R3(B) C3 W1(A) R1(B) W1(B) C1 W2(A) W2(B) C2

5. Si la stratégie wait-die est utilisée pour gérer les demandes de verrouillage. Quelle est l'exécution finale obtenue ?

Transaction	Action	Demande Verrou	Réponse
T1	R1(A)	SLOCK(A)	OK
T2	R2(A)	SLOCK(A)	Non, T2 mort (dies)
T2	R2(B)	SLOCK(B)	T2 is died
T3	R3(A)	SLOCK(A)	Non, T3 mort (dies)
T3	R3(B)	SLOCK(B)	T3 is died
T1	W1(A)	XLOCK(A)	OK
T1	R1(B)	SLOCK(B)	OK
T1	W1(B)	XLOCK(B)	OK, end of T1, UNLOCK(A), UNLOCK(B)
T2	W2(A)	XLOCK(A)	T2 is died
T2	W2(B)	XLOCK(B)	T2 is died
Relancer T2 depuis le début avec la même estampille 2 (R2(A) R2(B) W2(A) W2(B))			
Relancer T3 depuis le début avec la même estampille 3 (R3(A) R3(B))			

Exercice 04

L'ordonnancement O: R1(A) R2(A) R2(B) R3(A) R3(B) C3 W1(A) R1(B) W1(B) C1 W2(A) W2(B) C2

6. Si la stratégie wound-wait est utilisée pour traiter les demandes de verrouillage, dans quel ordre les transactions sont finalement validées?

Algorithme wound-wait

Soient 2 transactions T_i , T_j , un granule r utilisé par T_i et demandé par T_j , et soient les estampilles $E(T_i)$ et $E(T_j)$ associées aux deux transactions.

Algorithme: Wound- Die

Si $E(T_i) > E(T_j)$ alors annuler T_i /*Wound*/
sinon bloquer T_j /*Wait*/

Fsi

/* T_i est redémarrée ultérieurement avec la même estampille*/

- Dans la stratégie "wound-wait" une transaction plus jeune est autorisée à attendre une transaction plus ancienne, tandis qu'une transaction plus ancienne demandant un granule détenu par une transaction plus jeune tue la transaction plus jeune.

Exercice 03

L'ordonnancement O: R1(A) R2(A) R2(B) **R3(A) R3(B)** C3 W1(A) R1(B) W1(B) C1 W2(A) W2(B) C2

6. Si la stratégie wound-wait est utilisée pour traiter les demandes de verrouillage, dans quel ordre les transactions sont finalement validées?

Transaction	Action	Demande Verrou	Réponse
T1	R1(A)	SLOCK(A)	OK
T2	R2(A)	SLOCK(A)	Non, T2 attend (waits)
T2	R2(B)	SLOCK(B)	T2 is waiting
T3	R3(A)	SLOCK(A)	Non, T3 attend (waits)
T3	R3(B)	SLOCK(B)	T3 is waiting
T1	W1(A)	XLOCK(A)	OK
T1	R1(B)	SLOCK(B)	OK
T1	W1(B)	XLOCK(B)	OK, end of T1, UNLOCK(A), UNLOCK(B), wake up T2.
T2: exécuter R2(A) R2(B) W2(A) W3(B), UNLOCK(A), UNLOCK(B), wake up T3.			
T3: exécuter R3(A) R3(B), UNLOCK(A), UNLOCK(B)			

Exercice 05

- Un SGBD reçoit la séquence d'opérations suivante, en provenance de trois transactions :

O1: R1(X) R2(X) W2(Y) R3(Y) W3(Z) W1(Z) W2(X) R3(X) W3(X)

- L'un des programmes qui s'exécute dans ce SGBD gère les connexions à un système informatique. Ce programme lit le mot de passe de l'utilisateur, le compare avec celui stocké sous forme cryptée dans la base de données écrit les informations de connexion dans le journal de connexions et incrémente la valeur du nombre d'accès réalisés par l'utilisateur dans le système. Le journal de connexions et le nombre d'accès des utilisateurs sont stockés dans la base de données.

Exercice 05

- Un SGBD reçoit la séquence d'opérations suivante, en provenance de trois transactions :

O1: R1(X) R2(X) W2(Y) R3(Y) W3(Z) W1(Z) W2(X) R3(X) W3(X)

Pour l'ordonnancement O1 on a trois transactions T1, T2, T3 tel que:

- T1 : R1(X) W1(Z)
 - T2 : R2(X) W2(Y) W2(X)
 - T3 : R3(Y) W3(Z) R3(X) W3(X)
- T1 utilise **deux granules** X et Z et a **deux opérations** une lecture R1(X) et une écriture W1(Z).
- T2 utilise **deux granules** X et Y et a **trois opérations** une lecture R2(X) et deux écritures W2(Y), W2(X).
- T3 utilise **trois granules** X, Y et Z et a **quatre opérations** deux lectures R3(Y), R3(X) et deux écritures W3(Z), W3(X).

Exercice 05

□ L'un des programmes qui s'exécute dans ce SGBD gère les connexions à un système informatique. Ce programme **lit le mot de passe de l'utilisateur, le compare avec celui stocké sous forme cryptée dans la base de données, écrit les informations de connexion dans le journal de connexions et incrémente la valeur du nombre d'accès réalisés par l'utilisateur dans le système. Le journal de connexions et le nombre d'accès des utilisateurs sont stockés dans la base de données.**

- **lit le mot de passe de l'utilisateur:** lecture à partir de l'interface (entrée clavier).
- **le compare avec celui stocké sous forme cryptée dans la base de données:** lecture à partir de la BD: `select * from utilisateur where motdepasse=motdepasselu;` → **Read**.
- **écrit les informations de connexion dans le journal de connexions:** insertion d'une ligne dans la table journal → **Write**.
- **incrémente la valeur du nombre d'accès réalisés par l'utilisateur dans le système:** mise à jour de nombre d'accès → `update` → lecture de nombre d'accès et écriture (incrémentation) sur le nombre d'accès. → **Read** puis un **Write**.

Exercice 05

❑ L'un des programmes qui s'exécute dans ce SGBD gère les connexions à un système informatique. Ce programme **lit le mot de passe de l'utilisateur, le compare avec celui stocké sous forme cryptée dans la base de données, écrit les informations de connexion dans le journal de connexions et incrémente la valeur du nombre d'accès réalisés par l'utilisateur dans le système. Le journal de connexions et le nombre d'accès des utilisateurs sont stockés dans la base de données.**

❑ En conclusion

- le programme manipule trois granules: mot de passe, journal de connexion et le nombre d'accès.
- Le programme a 4 opérations: deux lectures et deux écritures.

Exercice 05

1. Est-ce que l'une des transactions ci-dessus peut représenter l'exécution de ce programme de gestion des connexions? Justifier votre réponse.

□ Le programme manipule trois granules: mot de passe, journal de connexion et le nombre d'accès et a 4 opération: deux lecture et deux écritures.

□ T3 utilise **trois granules** X, Y et Z et a **quatre opérations** deux lectures R3(Y), R3(X) et deux écritures W3(Z), W3(X).

→ D'où la transaction T3 qui représente le programme

Exercice 05

O1: R1(X) R2(X) W2(Y) R3(Y) W3(Z) W1(Z) W2(X) R3(X) W3(X)

2. Construire le graphe de précédence de cet ordonnancement. O1 est-il sérialisable ?

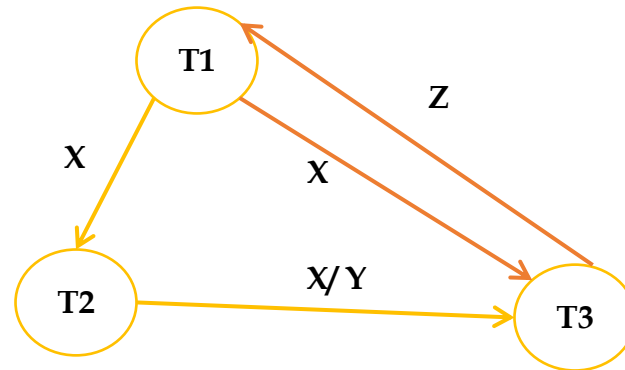
■ Les conflits :

sur X : R1(X)- W2(X), R1(X)- W3(X), R2(X)- W3(X), W2(X)-R3(X), W2(X)-W3(X)

sur Y : W2(Y)-R3(Y)

sur Z : W3(Z)-W1(Z)

■ Le graphe de précédence



■ Le graphe contient le cycle {T1, T3, T1}, l'ordonnancement n'est pas sérialisable.

Exercice 05

O1: R1(X) R2(X) W2(Y) R3(Y) W3(Z) W1(Z) W2(X) R3(X) **W3(X)**

3. Donner le scenario d'exécution de cet ordonnancement en appliquant le protocole de verrouillage à deux phases. Justifier.

Transaction	Action	Demande Verrou	Réponse
T1	R1(X)	SLOCK(X)	Ok
T2	R2(X)	SLOCK (X)	Ok
T2	W2(Y)	XLOCK (Y)	Ok
T3	R3(Y)	SLOCK(Y)	Non, T3 attend T2
T3	W3(Z)	XLOCK(Z)	T3 en attente de T2
T1	W1(Z)	XLOCK(Z)	Ok, fin de T1 d'où UNLOCK(X), UNLOCK(Z)
T2	W2(X)	XLOCK(X)	Ok, fin de T2 d'où UNLOCK(X), UNLOCK(Y) et réveiller T3
T3	R3(Y)	SLOCK(Y)	Ok
T3	W3(Z)	XLOCK(Z)	Ok
T3	R3(X)	SLOCK(X)	Ok
T3	W3(X)	XLOCK(X)	Ok, fin de T3 d'où UNLOCK(X), UNLOCK(Y), UNLOCK(Z)

Exercice 05

O1: R1(X) R2(X) W2(Y) R3(Y) W3(Z) **W1(Z)** W2(X) R3(X) W3(X)

4. L'algorithme d'estampillage (à deux estampilles) accepte-t-il cet ordonnancement sans rejets ?

Transaction	Action	EL(X)	EE(X)	EL(Y)	EE(Y)	EL(Z)	EE(Z)
		0	0	0	0	0	0
T1	R1(X)	1	0	0	0	0	0
T2	R2(X)	2	0	0	0	0	0
T2	W2(Y)	2	0	0	2	0	0
T3	R3(Y)	2	0	3	2	0	0
T3	W3(Z)	2	0	3	2	0	3
T1	W1(Z)	EE(Z)=3>1, T1 est annulée et sera relancée à la fin					

→ D'où l'algorithme s'exécute avec un rejet

Exercice 05

O2: R1(X) R2(X) R3(Y) W2(Y) W3(Z) W1(Z) R3(X) W2(X) W3(X).

5. Construire le graphe d'attente de cet ordonnancement. Existe-t-il un deadlock ? Si oui, proposer une solution à ce problème.

a. V2P

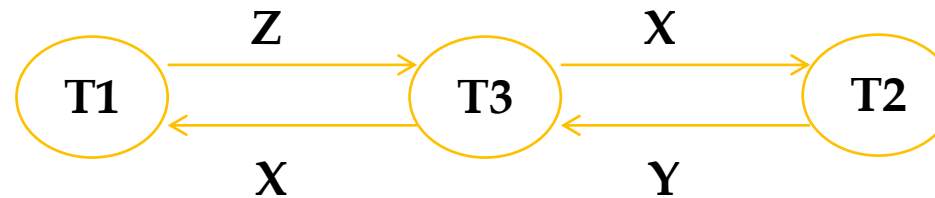
Transaction	Action	Demande Verrou	Réponse
T1	R1(X)	SLOCK(X)	Ok
T2	R2(X)	SLOCK (X)	Ok
T3	R3(Y)	SLOCK (Y)	Ok
T2	W2(Y)	XLOCK (Y)	Non, T2 attend T3
T3	W3(Z)	XLOCK(Z)	Ok
T1	W1(Z)	XLOCK(Z)	Non, T1 attend T3
T3	R3(X)	SLOCK(X)	Ok
T2	W2(X)	XLOCK(X)	T2 en attente de T3
T3	W3(X)	XLOCK(X)	Non, T3 attend T1 et T2

Exercice 05

O2: R1(X) R2(X) R3(Y) W2(Y) W3(Z) W1(Z) R3(X) W2(X) W3(X).

5. Construire le graphe d'attente de cet ordonnancement. Existe-t-il un deadlock ?
Si oui, proposer une solution à ce problème.

b. Graphe d'attente



✓ De graphe d'attente, on constate qu'on a deux cycles $C1=\{T1-T3-T1\}$ et $C2=\{T2-T3-T2\}$ d'où on a un inter-blockage. $C1 \cap C2 = \{T3\}$

c. Solution : T3 est la victime. **Rollback(T3)** → annuler T3(undo(W3(Z)) et libérer les verrous(unlock(Y); Unlock(Z)) et **reprendre** T2 (la plus ancienne) à partir de point de blockage (FIFO).

Exercice 05

O2: R1(X) R2(X) R3(Y) W2(Y) W3(Z) W1(Z) R3(X) W2(X) W3(X).

5. Construire le graphe d'attente de cet ordonnancement. Existe-t-il un deadlock ? Si oui, proposer une solution à ce problème.

a. V2P

Transaction	Action	Demande Verrou	Réponse
Rollback(T3): UNDO(W3(Z)), UNLOCK(X), UNLOCK(Y), UNLOCK(Z) et reprise de T2			
T2	W2(Y)	XLOCK(Y)	OK, fin de T2 d'où UNLOCK(X), UNLOCK(Z)
T2	W2(X)	XLOCK(X)	Non, T2 attend la fin de T1
Reprise de T1			
T1	W1(Z)	XLOCK(Z)	OK, fin de T1, UNLOCK(X), UNLOCK(Z) , réveiller T2
T2	W2(X)	XLOCK(X)	OK, fin de T2 d'où UNLOCK(X), UNLOCK(Y)
Relancer T3 dès son début : T3: R3(Y) W3(Z) R3(X) W3(X)			