Gestion des transactions / de la concurrence

Plan

1. Notion de transaction
2. Vie d’une transaction
3. Gestion des transactions
4. Concurrence d’accès
5. Verrouillage
6. Sérialisabilité
7. Protocole de verrouillage à 2 phases
8. Niveau d’isolation

1. Notion de transaction

- Unité logique de traitement qui est :
  - soit complètement exécutée
  - soit complètement abandonnée
- Une transaction est une unité atomique de traitement
- Une transaction fait passer la base de données d'un état cohérent à un autre état cohérent
- Si une transaction ne va pas à son terme pour une raison ou pour une autre, la base est restaurée dans l'état où elle se trouvait avant que la transaction ne démarre

1. Notion de transaction

Exemple : le transfert d'une somme S d'un compte C1 vers un compte C2

(1) début-transaction
(2) lire C1
(3) C1 := C1 - S
(4) écrire C1
(5) lire C2
(6) C2 := C2 + S
(7) écrire C2
(8) fin-transaction

Cette transaction est constituée d'un ensemble d'actions élémentaires, mais elle doit être traitée comme une seule opération.

Autrement dit le gestionnaire des transactions doit assurer que **toutes les actions de la transaction sont exécutées, ou bien** qu'**aucune ne l'est.**

1. Notion de transaction

Un système de gestion transactionnel doit garantir les propriétés suivantes :

- ***Atomicité***

  Une transaction doit effectuer toutes ses mises à jour ou rien faire du tout

- ***Cohérence***

  La transaction doit faire passer la base de données d'un état cohérent à un autre

- ***Isolation***

  Les résultats d'une transaction ne doivent être visibles aux autres transactions qu'une fois la transaction validée

- ***Durabilité***

  Dès qu'une transaction valide ses modifications, le système doit garantir que ces modifications seront conservées en cas de panne

2. Vie d'une transaction

♦ **Vie sans histoire**

- La transaction s'exécute normalement jusqu'à la fin. Elle se termine par une instruction de validation **COMMIT** en SQL (transaction *validée).*
- Toutes les modifications faites sur la base par cette transaction sont considérées comme définitives.

♦ **Un assassinat**

- Un événement extérieur vient interrompre l'exécution de la transaction de façon irrémédiable
- Cet arrêt peut provenir, soit d'une panne, soit d'une action délibérée de la part du SGBD qui décide de supprimer telle ou telle transaction (c'est le cas lorsqu'il détecte un interblocage).

2. Vie d’une transaction

♦ **Un suicide**

- Au cours de son exécution la transaction détecte certaines conditions qui font que la poursuite de son exécution s'avère impossible, elle peut se supprimer en exécutant une instruction d'annulation **ROLLBACK** en SQL
- Dans ces deux derniers cas, tout doit se passer comme si la transaction n'avait jamais existée. Il faut donc en quelque sorte lui faire faire marche arrière et effacer de la base de données toute trace de son exécution : nous dirons que la transaction a été *annulée.*

3. Gestion des transactions

Une transaction est implémentée par :

- TID : Identificateur de transaction
- Inscription dans la BD seulement après le COMMIT
- Journalisation:
  - Toute opération d'une transaction est notée avant et après l'exécution dans un fichier journal présumé à l'abris de pannes
  - Les opérations de commitement sont notées avant d'être exécutées sur la BD (write- ahead log protocol)
- Points de reprise (checkpoints)
  - sauvegardes de l'état de la BD et notamment de TIDs de transactions en cours à intervalles réguliers

3. Gestion des transactions

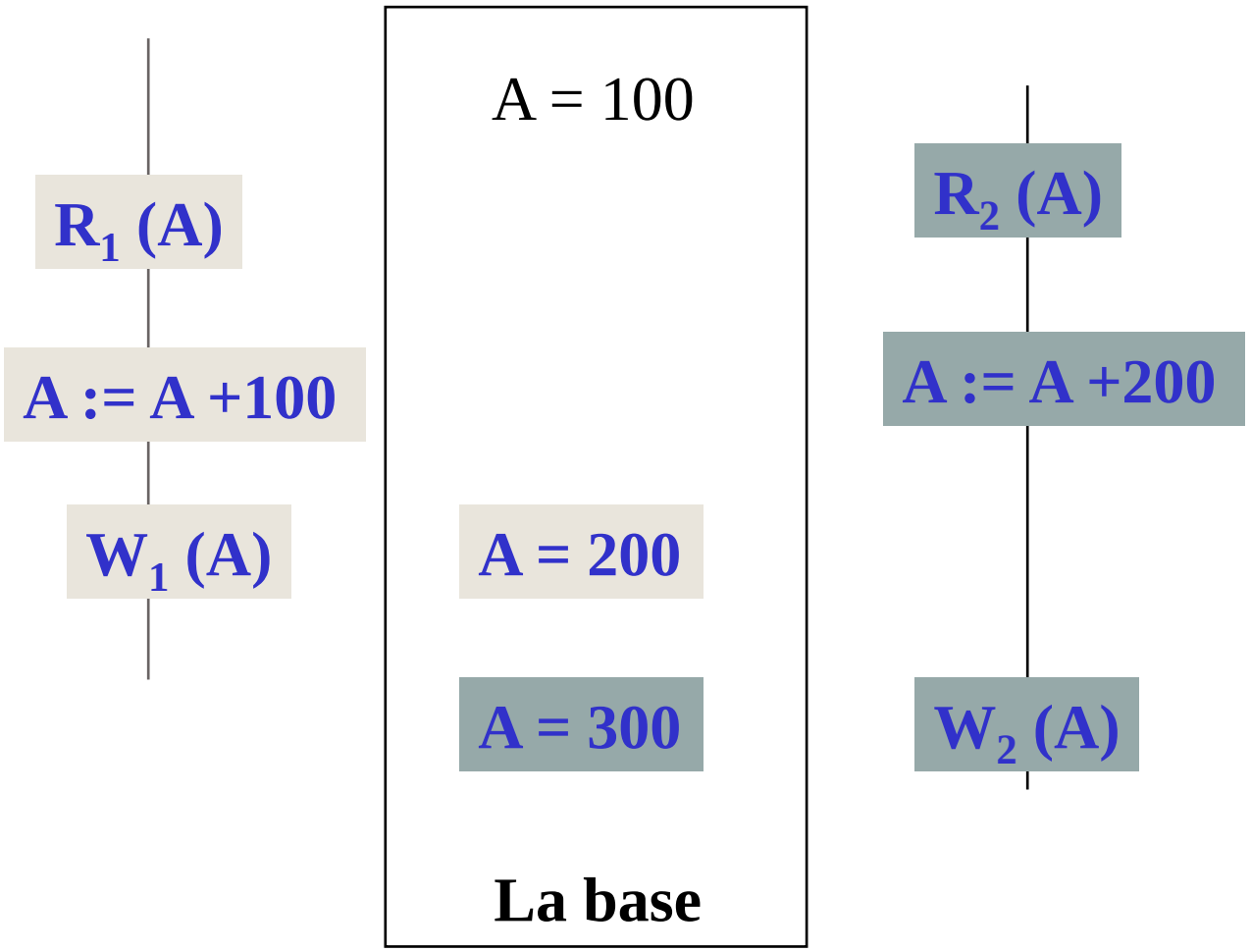
En cas de problème :

- On ferme l'accès à la base
- On reprend le dernier checkpoint
- On retrouve sur le journal toutes les transactions commises après
  - commencées avant ou après le checkpoint
- On reexécute chronologiquement ces transactions
  - et seulement ces transactions
- On rouvre la base aux usagers

4. Concurrence d'accès

- Dans un contexte multiutilisateurs, les BDs étant partagées, les transactions pourraient être exécutées:
  - l'une après l'autre (séquentiellement)
  - Simultanément (parallèlement)
    (+) meilleures performances
    (-) possibilités d'inconsistances dans la base
- Théorie de concurrence analyse les problèmes d'accès simultané

4. Concurrence d'accès



4. Concurrence d'accès

- Des transactions exécutées concurremment peuvent interférer et mettre la base de données dans un état incohérent.
- Considérons deux transactions T1 et T2 qui s'intéressent à un même objet A
- Les deux seules opérations possibles sur A, sont : lire et écrire

Quatre possibilités :

A. LECTURE-LECTURE ET PARTAGE

- Aucun conflit
- Un même objet peut toujours être partagé en lecture

4. Concurrence d'accès

B. ECRITURE-ECRITURE ET PERTE DE MISE A JOUR

T2 vient "écraser" par une autre écriture celle effectuée par T1

Temps	Transaction T1	Etat de la base	Transaction T2
t1	lire A	A = 10	-
t2	-		lire A
t3	A := A + 10		-
t4	-		-
t5	-		A := A + 50
t6	écrire A	A = 20	-
t7	-	A = 60	écrire A

4. Concurrence d'accès

C. ECRITURE-LECTURE ET LECTURES IMPROPRES

T2 lit une valeur modifiée par T1 et ensuite T1 est annulée

Temps	Transaction T1	Etat de la base	Transaction T2
t1	lire A	A=10	-
t2	A := A+ 20		-
t3	écrire A	A = 30	-
t4	-		lire A
t5	**annulation**		
t6	-		-

4. Concurrence d'accès

D. LECTURE-ECRITURE ET LECTURES NON REPRODUCTIBLES

T1 modifie la valeur de A entre deux lectures de T2

Temps	Transaction T1	Etat de la base	Transaction T2
t1	lire A	A=10	-
t2	-		lire A
t3	A := A + 10		-
t4	écrire A	A = 20	-
t5	-		-
t6	-		lire A

De nombreuses solutions ont été proposées pour traiter le problème des accès concurrents.

5. Verrouillage

- C'est la technique la plus classique pour éviter les problèmes dus à la concurrence :
  - chaque transaction verrouille les données qu'elle lit ou écrit pour interdire aux autres transactions d'y accéder.
- Le verrouillage dégrade les performances d'un SGBD en imposant des temps d'attente :
  - il y a donc intérêt à limiter la taille des données à verrouiller, d'où le concept de **granularité de verrouillage.**
- On peut encore améliorer les performances en précisant la nature des opérations pour lesquelles le verrouillage est réalisé (lecture ou écriture) :
  - cela conduit à la définition de **modes de verrouillage.**

Granularité

- On peut verrouiller :
  - une valeur d’attribut,
  - un n-uplet et donc toutes ses valeurs,
  - une page de fichier et donc tous ses n-uplets,
  - une table et donc toutes ses lignes,
  - la BD et donc toutes ses tables

5. Verrouillage

Il repose sur les deux actions :

- **Verrouiller (A) : acquérir un contrôle de l'objet A**
- **Libérer (A) : libérer l'objet A**

Un objet A est typiquement un n-uplet de la BD

- **Il y a deux types de verrous :**

  - ♦ Verrous **exclusifs (X locks)** ou verrous d'écriture
  - ♦ Verrous **partagés (S locks)** ou verrous de lecture

5. Verrouillage

Protocole d'accès aux données

1. Aucune transaction ne peut effectuer une lecture ou une mise à jour d'un objet si elle n'a pas acquis au préalable un verrou S ou X sur cet objet
2. Si une transaction ne peut obtenir un verrou déjà détenu par une autre transaction T2, alors elle doit attendre jusqu'à ce que le verrou soit libéré par T2
3. Les verrous X sont conservés jusqu'à la fin de la transaction (COMMIT ou ROLLBACK)
4. *En général les verrous S sont également conservés jusqu'à cette date*

5. Verrouillage

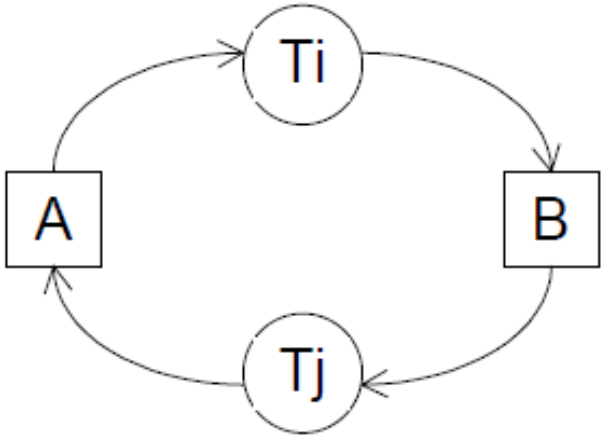
Phénomènes indésirables

La privation

- Une transaction risque d'attendre un objet indéfiniment si à chaque fois que cet objet est libéré, il est pris par une autre transaction.
- Pour traiter ce problème, on peut organiser sur chaque verrou une file d'attente avec une politique «première arrivée» , « première servie ».

5. Verrouillage

L'interblocage (ou verrou mortel)



Ti attend Tj , Tj attend Ti : il y a interblocage

On peut construire le graphe « Qui attend Quoi » :
- les sommets représentent les transactions Ti
- on aura une arête Ti → Tj si Ti est en attente de Tj

Il y a situation d'interblocage lorsque le graphe contient un cycle.

5. Verrouillage

L'interblocage (Exemple)

Interblocage entre deux transactions

Base : **Modules** (Sigle, Durée, Mat)

Transactions :

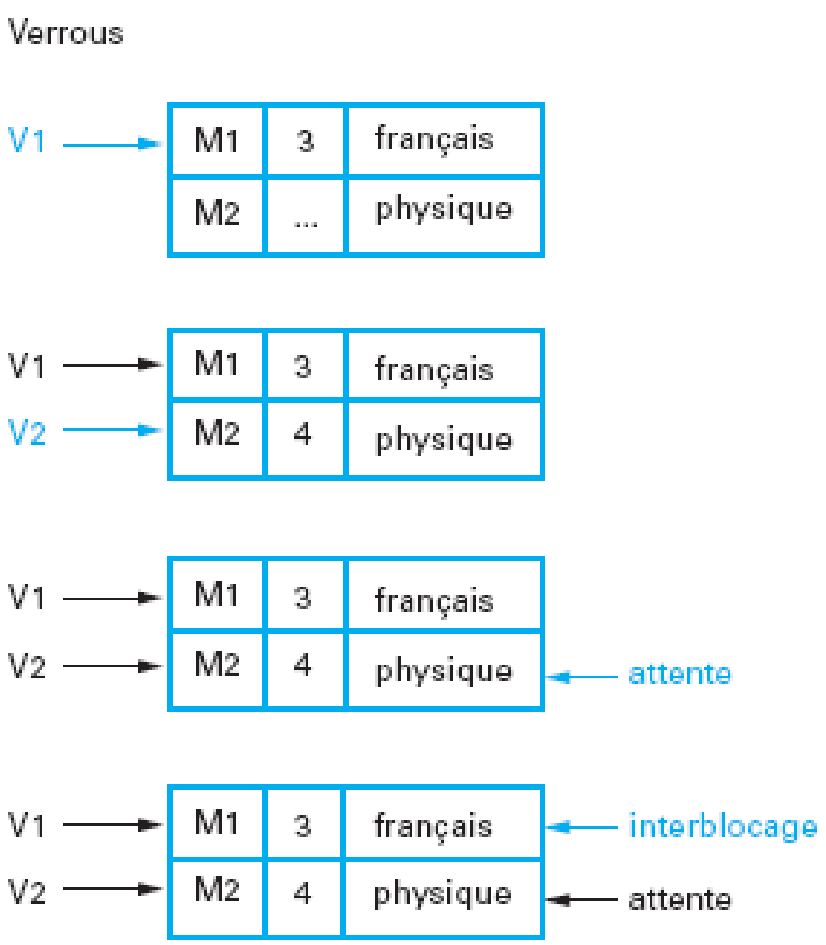
T1 : UPDATE Modules SET durée = 3
WHERE sigle = 'M1'

T2 : UPDATE Modules SET durée = 4
WHERE sigle = 'M2 '

T1 : UPDATE Modules SET durée = 4
WHERE sigle = 'M2 '

T2 : UPDATE Modules SET durée = 3
WHERE sigle = 'M1 '

temps



5. Verrouillage

2 techniques pour traiter le problème d'interblocage :

Prévention des interblocages

- Lorsqu'une demande d'acquisition de verrou ne peut être satisfaite, on fait passer un test aux deux transactions impliquées à savoir celle qui demande le verrou, Ti, et celle qui le possède déjà, Tj.
- Si Ti et Tj passent le test alors Ti est autorisé à attendre Tj, sinon l'une des deux transactions est annulée pour être relancée par la suite.

Détection des interblocages

5. Verrouillage

2 techniques pour traiter le problème d'interblocage :

Prévention des interblocages

Détection des interblocages

- Les interblocages sont détectés en construisant effectivement le graphe « qui attend quoi » et en y recherchant les cycles.
- Lorsqu'un cycle est découvert l'une des transactions est choisie comme victime, elle est annulée de manière à faire disparaître le cycle.

6. Sérialisabilité

- Le principe sur lequel repose le contrôle de concurrence est celui de la **sérialisabilité** de l'exécution d'un ensemble de transactions.
- En effet, lorsque les transactions sont exécutées les unes après les autres, il n'y a pas de problèmes de concurrence.
- Cette solution est inapplicable car très coûteuse en temps calcul.
- La solution adoptée consiste à exécuter un ensemble de transactions concurrentes de façon à ce que le résultat soit équivalent à une exécution en série de ces transactions : une telle exécution est dite **sérialisable**.

6. Sérialisabilité

Une exécution entrelacée d'un ensemble de transactions est considérée correcte si elle est sérialisable, c'est-à-dire, si elle produit le même résultat qu'une certaine exécution *en série des mêmes* transactions.

Ordonnancement

- Etant donné un ensemble de transactions, toute exécution de ces transactions (entrelacée ou non) est appelé un ordonnancement.

Théorème de verrouillage à deux phases :

- Si toutes les transactions satisfont le *« protocole de verrouillage à deux phases », tous les ordonnancements* entrelacés sont alors sérialisables.

7. Protocole de verrouillage à 2 phases

- Pour chaque transaction, tous les verrouillages doivent précéder toutes les libérations de verrous.
- Après l'abandon d'un verrou, une transaction ne doit plus jamais pouvoir obtenir de verrous. On distingue deux phases :
  - **Acquisition des verrous**
  - **Libération des verrous.**
- Dans la pratique, la seconde phase est souvent condensée en une seule opération de COMMIT ou de ROLLBACK à la fin de la transaction.
- Dans le but de réduire les conflits sur les ressources et améliorer les performances, les systèmes réels autorisent la construction de transactions qui ne sont pas à deux phases c'est-à-dire qui abandonnent prématurément des verrous (avant le COMMIT) et obtiennent ensuite de nouveaux verrous.

7. Protocole de verrouillage à 2 phases

- Plus de perte de mise à jour :

T1	Etat de la BD	T2
Verrouiller A		
Lire A	A=10	
		Verrouiller A
A := A+10		Attente
Ecrire A	A = 20	Attente
Libérer A		Attente
	A = 20	Lire A
		A := A+50
	A = 70	Ecrire A
		Libérer A

7. Protocole de verrouillage à 2 phases

- Plus de lecture impropre :

T1	Etat de la BD	T2
Verrouiller A		
Lire A	A=10	
A := A+20		
Ecrire A	A = 30	
		Verrouiller A
Annulation	A = 10	Attente
Libérer A		Attente
	A = 10	Lire A

7. Protocole de verrouillage à 2 phases

- Plus de lectures non reproductibles :

T1	Etat de la BD	T2
Verrouiller A		
Lire A	A=10	
		Verrouiller A
A := A+10		Attente
Ecrire A	A = 20	Attente
Libérer A		Attente
	A = 20	Lire A
	A = 20	Lire A
		Libérer A

7. Protocole de verrouillage à 2 phases

- Plus de lectures non reproductibles :

T1	Etat de la BD	T2
		Verrouiller A
	A=10	Lire A
Verrouiller A		
Attente	A = 10	Lire A
Attente		Libérer A
Lire A	A = 10	
A := A+10		
Ecrire A	A = 20	
Libérer A		

8. Niveau d'isolation

- Tout protocole qui n'est pas complètement sérialisable ne peut être considéré sûr, cependant, les systèmes autorisent des transactions s'exécutant à un **niveau d'isolation non sûr qui pourrait violer la sérialisabilité de** trois façons particulières :

A. Lecture salissante

- Supposons que la transaction T1 effectue une mise à jour sur une certaine ligne, que la transaction T2 récupère ensuite cette ligne et que la transaction T1 se termine par un ROLLBACK.
- La transaction T2 a alors observé une ligne qui n'existe plus, et dans un certain sens n'a jamais existé (car la transaction T1 n'a en fait jamais été exécutée).

8. Niveau d'isolation

B. Lecture non renouvelable

- Supposons que la transaction T1 récupère une ligne, que la transaction T2 effectue ensuite une mise à jour de cette ligne et que la transaction T1 récupère de nouveau la « même » ligne. La transaction T1 a en fait récupéré la « même » ligne deux fois mais a observé des valeurs différentes de cette ligne.

C. Fantômes

- Supposons que la transaction T1 récupère un ensemble de lignes qui satisfont une certaine condition. Supposons que la transaction T2 insère ensuite une ligne qui satisfait la même condition. Si la transaction T1 répète maintenant la même demande, elle observera une ligne qui n'existait pas précédemment – un « fantôme ».

8. Niveau d'isolation

Objets Fantômes (Exemple – Problème)

T1	T2
SELECT COUNT(*) FROM livre WHERE annee = 2013; **(réponse *n*)** SELECT COUNT(*) FROM livre WHERE annee = 2013; **(réponse *n* + *1*)**	INSERT INTO livre VALUES ("Les BD", 2013);

8. Niveau d'isolation

Objets Fantômes (Exemple – Solution)

T1	T2
Lock S livre SELECT COUNT(*) FROM livre WHERE annee = 2013; **(réponse *n)*** SELECT COUNT(*) FROM livre WHERE annee = 2013; **(réponse *n)*** *Unlock livre*	Lock X livre Attente Attente Attente Attente Attente Attente Attente INSERT INTO livre VALUES ("Les BD", 2013);

8. Niveau d'isolation

Niveaux d'isolation SQL

- L'instruction **SET TRANSACTION** permet de définir le mode d'exécution (qui inclut le mode d'accès et le niveau d'isolation) de la prochaine transaction à exécuter
- SET TRANSACTION (*mode d'accès* | *niveau d'isolation)*
  - *mode d'accès* ➔ *READ ONLY* | *READ WRITE*
  - *niveau d'isolation* ➔ READ UNCOMMITED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE
- L'option par défaut est SERIALIZABLE

- Attention ! la commande SET TRANSACTION n'est pas un début de transaction et elle ne peut pas être utilisée si une transaction est en cours.

8. Niveau d'isolation

Niveau d'isolation	lecture salissante	lecture non renouvelable	fantôme
READ UNCOMMITTED Lecture non validée	O	O	O
READ COMMITTED Lecture validée	N	O	O
REPEATABLE READ Lecture renouvelable	N	N	O
SERIALIZABLE Sérialisable	N	N	N