I Gestion de transactions

La 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

Exemple du banquier :

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.

La 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

Nous dirons que cette transaction est *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).

♦ 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*.

La gestion des transactions

Un système de gestion transactionnel doit garantir les propriétés suivantes (résumées par le vocable **ACID**) :

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

Les problèmes de 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 :

1. LECTURE-LECTURE ET PARTAGE

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

2. 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

3. 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. 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.

- Un exemple important est le protocole appelé *verrouillage à deux phases* qui est un des plus utilisés.

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

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

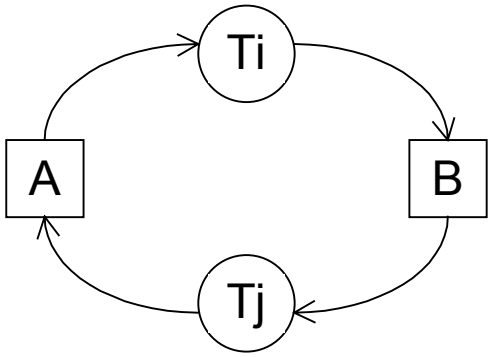
Phénomèmes indésirables

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".

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.

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 posséde 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

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.

L'examen du graphe peut se faire

- soit lorsqu'il y a attente de la part d'une transaction,
- soit périodiquement

Sérialisabilité

Une exécution entrelacée donnée 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 s'exécutant l'une après l'autre

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.

Protocole de verrouillage à deux 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, par la même, d'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.

Niveaux d'isolation

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

❖ 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).

❖ 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 récupérer la « même » ligne deux fois mais a observé des valeurs différentes de cette ligne.

❖ 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 ».

Niveaux d'isolation SQL

L'instruction **SET TRANSACTION** permet de définir le *niveau d'isolation* de la prochaine transaction à exécuter

SET TRANSACTION READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE

- L'option par défaut est SERIALIZABLE

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