Chapitre IV : Concurrence d'accès

Introduction

Durant le fonctionnement d'une base de données, il est fréquent que des requêtes lancées par des utilisateurs ou des applications veuillent accéder simultanément aux ressources (CPU, RAM, etc.). Sur un SGBD on suppose que chaque tâche lancée est intégralement exécutée indépendamment des autres et à chaque fois qu'elle est appelée. Il est, cependant, fréquent que l'exécution d'une tâche soit interrompue. Le SGBD doit, néanmoins, veiller à une bonne exécution des demandes dans le bon ordre tout en assurant la cohérence de la base et la disponibilité des données.

Les SGBD relationnels implémentent des techniques permettant d'assurer un partage concurrent des données et une gestion des arrêts spontanés d'exécution. Ainsi, un programme se comporte, au moment de son exécution, comme s'il était le seul à accéder aux ressources. La technique implémentée par les SGBD pour assurer cela est appelée ***contrôle de concurrence***.

I. Transaction

I. 1. Qu'est-ce qu'une transaction ?

Une transaction est une séquence d'opérations regroupées dans un programme exécutée par un SGBD sur une base de données et intégralement validée par un ***Commit*** ou totalement annulée par un ***Rollback***. Les opérations constituant une transaction forment une unité d'exécution (toutes les opérations sont exécutées ou aucune ne l'est). La succession des opérations d'une transaction doit être cohérente tout en ayant un sens.

I. 2. Quelques définitions

Exécution sérialisable : Une exécution en parallèle de n transactions T_1, T_2, …, T_n est dite sérialisable si son résultat est équivalent à une exécution en série quelconque de ces mêmes transactions.

Opérations commutables : Les opérations A et B sont commutables (permutables) si l'exécution de A suivie de B est équivalente (donne le même résultat) à celle de B suivie de A.

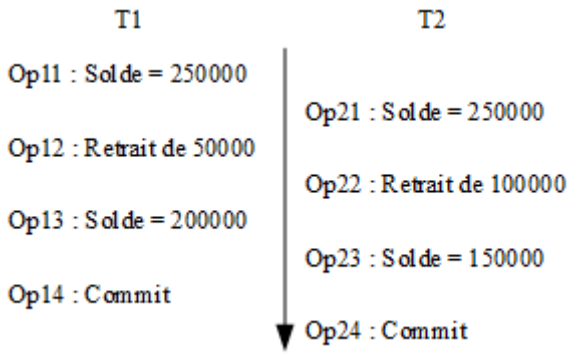
Si une exécution de transactions est transformable par commutations successives en une exécution en série alors elle est sérialisable.

Concurrence parfaite : Plusieurs transactions peuvent être exécutées simultanément par un serveur de données. Le SGBD n'exécute pas les transactions en séquence (les unes après les autres) mais en parallèle. Il est alors possible qu'il y ait entrelacement des opérations de différentes transactions. Si chaque opération est exécutée dès son arrivée, on parle de concurrence parfaite.

Il peut survenir un problème lors de l'exécution des opérations. Il faut alors un contrôle de concurrence pour résoudre ce problème. Pour cela le système modifie l'entrelacement des opérations pour avoir une exécution correcte des transactions. Ainsi, le système retarde l'exécution de certaines opérations. Cependant, ce procédé peut impliquer des pertes de performance.

II. Perte de mise à jour

Soit un compte C_1 avec un solde de 250 000. Au même moment deux transactions T_1 et T_2 sont lancées dont l'une fait un retrait de 50 000 et l'autre un retrait de 100 000.



Si les transactions T_1 et T_2 s'exécutent comme ci-dessus, il y a perte de mise à jour. A la fin, le solde est de 150 000 au lieu de 100 000.

Il faut aussi noter que malgré la prise en compte par les SGBD relationnels de la résolution des accès concurrents, il faut prévoir, lors de l'écriture d'un programme, un certain nombre de mécanismes permettant de résoudre les problèmes qui peuvent survenir dans certains cas. Parmi ces mécanismes, nous avons :

III. Mécanismes de gestion des accès concurrents

III. 1. Points de sauvegarde

Il s'agit de mettre dans un programme un mécanisme permettant de garder l'état cohérent de la base de données à des moments donnés de son exécution. Ceci permet de rétablir la

cohérence de la base de données avec l'état cohérent le plus proche possible du point d'interruption.

Ces points de sauvegarde permettant d'annuler une partie de la transaction sans annuler toutes les opérations déjà exécutées.

Syntaxe :

Debut transaction

Instruction 1 ;

Instruction 2 ;

Savepoint t1 ;

Instruction 3 ;

Savepoint t2 ;

Instruction 4 ;

Instruction 5 ;

RollBack to t2 ; -- permet d'annuler les instructions 4 et 5

RollBack to t1 ; -- permet d'annuler les instructions 3, 4 et 5

RollBack ; -- permet d'annuler la transaction dans son intégralité

Fin transaction

III. 2. Blocage des autres utilisateurs

Il s'agit de verrouiller des ressources (tables, enregistrements). Ce mécanisme permet de bloquer les utilisateurs voulant accéder à ces ressources momentanément.

III. 3. Niveau d'isolation

Il s'agit de mettre en place un mécanisme permettant d'avoir plus de concurrences. Pour cela, il faut demander au système d'avoir plus de fluidité dans les exécutions concurrentes en mettant un niveau d'isolation moins strict. Il faut alors prendre en charge dans son programme le verrouillage des ressources critiques.

III. 4. Les verrous

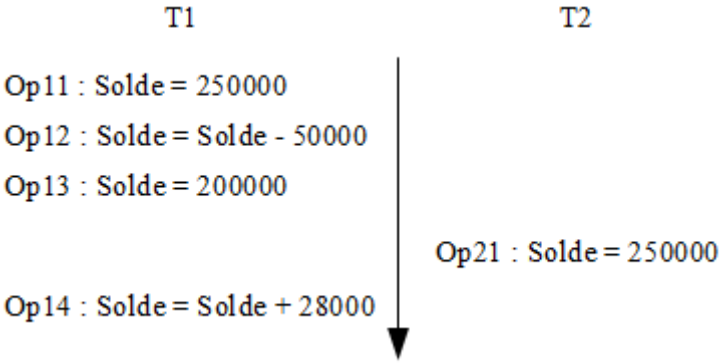
III. 4. 1. Verrous implicites

Les verrous implicites sont posés par le système pour résoudre le problème des accès concurrents et assurer la cohérence des données selon les trois règles suivantes :

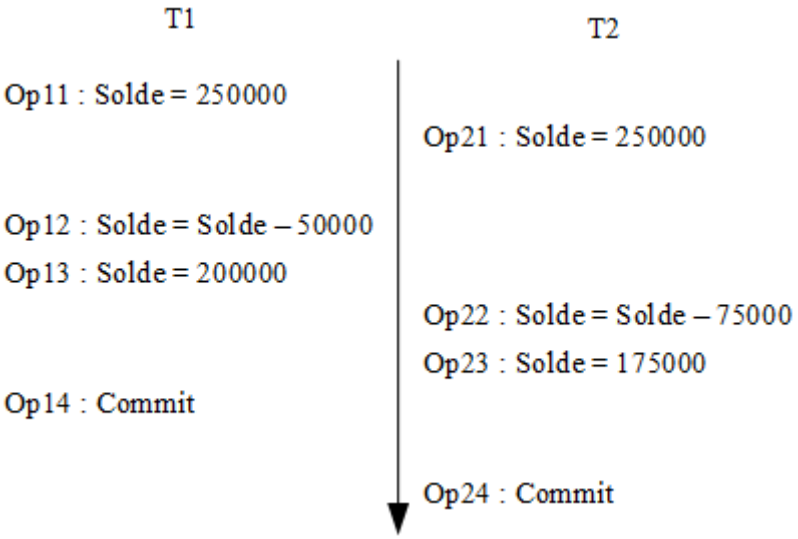
1. Une modification bloque toute autre modification sur les mêmes données ;
2. Une lecture ne bloque pas une modification sur les mêmes donnés ;
3. Une modification ne bloque pas une lecture sur les mêmes donnés.

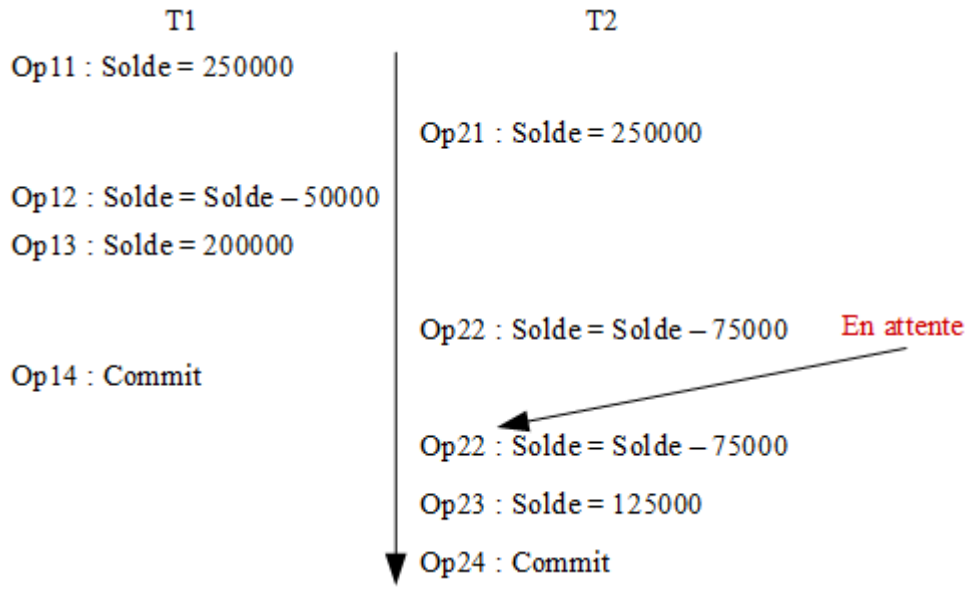
Ainsi, les verrous implicites tels que implémentés par Oracle garantit la cohérence en lecture au niveau opération : pas de lecture impropre et pas de perte de mise à jour.

- ✓ **Lecture impropre :** Il y a lecture impropre si une transaction lit des données écrites par une transaction concurrente non validée.



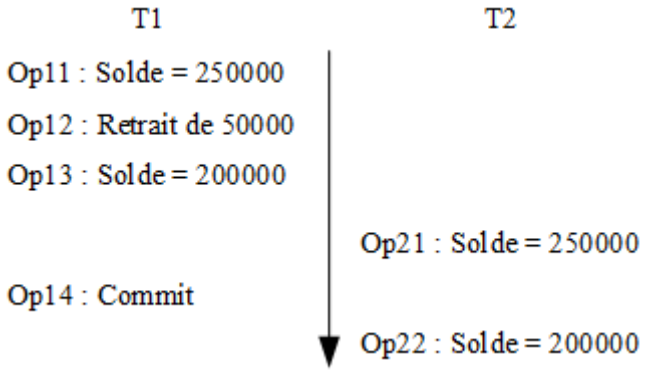
- ✓ **Perte de mise à jour :** Il y a perte de MAJ si une modification sur une donnée est écrasée par une autre transaction.





Cependant Oracle ne garantit pas la cohérence en lecture au niveau transaction, il y a alors possibilité de :

- ✓ **Références fantômes :** une transaction ré-exécute une requête et obtient un ensemble d'enregistrements différents (à cause d'une transaction concurrente validée)
- ✓ **Lectures non reproductibles :** Si une transaction relit des données et obtient des données modifiées (à cause d'une transaction concurrente validée)



Pour assurer la cohérence des lectures dans une transaction, on peut utiliser des verrous explicites.

III. 4. 2. Verrous explicites

Sous Oracle il existe deux méthodes de verrouillage explicite des données :

- ➢ **Verrouillage explicite de lignes :** Ce type de verrouillage permet de verrouiller des lignes répondant à une condition. Les lignes verrouillées ne peuvent être mises à jour que par la transaction bloquante. Toutes les autres lignes de la table sont toujours

disponibles pour mise à jour par d'autres transactions. Bien entendu, les autres sessions peuvent continuer de lire la table, y compris la ligne qui est en cours de mise à jour. Lorsque d'autres sessions lisent des lignes mises à jour, ces sessions ne peuvent que lire la version ancienne de la ligne avant mise à jour jusqu'à ce que les changements soient effectivement validés par la transaction bloquante.

Lorsqu'une transaction place un verrou ligne sur un enregistrement :

- un verrou de manipulation de données est placé sur la ligne. Ce verrou empêche les autres sessions de manipuler ou verrouiller cette ligne. Le verrou est relâché seulement lorsque la transaction bloquante est annulée ou validée.
- un verrou DDL est placé sur la table pour empêcher les altérations de structure de la table. Ce verrou est relâché également seulement lorsque la transaction bloquante annulée ou validée.

Select *|Liste, attributs From Table Where Condition **FOR UPDATE** ;

- **Verrouillage explicite de table :**

LOCK Table Table1[, Table2, …] **IN** mode_verrou **MODE** ;

Les modes de verrouille possibles pour une table sont :

- ✓ **Row Share (lignes partagées) :** Permet de verrouiller des lignes sélectionnées d'une table en vue de leur modification.
  - Toutes les opérations et tous les autres types de verrouillage à l'exception du verrouillage exclusif de table sont autorisés.

LOCK Table Table1[, Table2, …] **IN Row Share MODE** ;

Select *|Liste attributs From Table Where Condition **FOR UPDATE** ;

- ✓ **Row Exclusive (lignes exclusives) :** Permet de verrouiller des lignes sélectionnées d'une table en vue de leur modification. Si ce mode est activé :
  - Toutes les opérations et tous les autres types de verrouillage à l'exception des verrouillages exclusif de table, table partagée et ligne partagée sont autorisés.
  - C'est ce type de verrouillage qui est appliqué si on lance les requêtes suivantes :

Insert Into Table … ;

Update Table Set … ;

Delete From Table …;

LOCK Table Table1[, Table2, …] **IN Row Exclusive MODE** ;

- ✓ **Share (tables partagées) :** Si ce mode est activé :
  - les autres transactions peuvent effectuer des sélections sur cette table ;
  - Elles peuvent également poser les verrous en mode lignes partagées ou tables partagées ;
  - Toute opération apportant une modification à la table est impossible.

LOCK Table Table1[, Table2, …] **IN Share MODE** ;

- ✓ **Share Row Exclusive (partage exclusif de lignes) :** Ce mode de verrouillage permet de verrouiller une table entière. Si ce mode est activé :
  - Les autres transactions peuvent effectuer des sélections sur cette table. Elles peuvent également poser les verrous en mode lignes partagées.
  - Toute opération apportant une modification à la table est impossible. Une seule transaction peut poser à un moment donné ce mode de verrou sur une table.

LOCK Table Table1[, Table2, …] **IN Share Row Exclusive MODE** ;

- ✓ **Exclusive (tables exclusives) :** Ce mode de verrouillage permet de verrouiller une table entière. Si ce mode est activé :
  - Les autres transactions ne peuvent effectuer que des sélections sur cette table ;
  - Toute opération apportant une modification à la table est impossible ;
  - Une seule transaction peut poser à un moment donné ce mode de verrou sur une table.

LOCK Table Table1[, Table2, …] **IN Exclusive MODE** ;

III. 4. 3. Compatibilité entre les modes de verrouillage

Mode de verrouillage		**Compatibilité avec les autres modes de vérrouillage**				
		1	**2**	**3**	**4**	**5**
Lignes partagées	**1**	Ok	Ok	Ok	Ok	
Lignes exclusives	**2**	Ok	Ok			
Tables partagées	**3**	Ok		Ok		
Partage exclusif de lignes	**4**	Ok				
Tables exclusives	**5**					

IV. Exemple

1. Soit le schéma de base de données suivant :

Client (Id, Nb_Places_Reservees, Prix_A_Payer)

Spectacle (Id, Nb_Places, Nb_Places_Libres, Tarif)

Ecrivez une transaction qui gère les réservations faites par les clients pour assister à des spectacles. Chaque réservation est faite par un client. Un spectacle a un nombre de places donné. Un client peut alors réserver plusieurs places pour un spectacle.

```
Create Or Replace Procedure Reservation (p_idCl Int, p_idSp Int, p_nbP Int) AS
    cl                  Client%ROWTYPE ;
    sp                  Spectacle%ROWTYPE ;
    places_libres       Int ;
    places_reservees    Int ;
    somme_a_payer       Int ;
    prix_act            Int ;
Begin
    Select * Into sp From Spectacle Where id = p_idSp ;
    If (sp.nb_places_libres >= p_nbP) Then
        Select * Into cl From Client Where id = p_idCl ;
        places_libres := sp.nb_places_libres – p_nbP ;
        places_reservees := cl.nb_places_reservees + p_nbP ;
        Update Spectacle Set nb_places_libres = places_libres Where id = p_idSp ;
        Update Client Set nb_places_reservees = places_reservees
        Where id = p_idCl ;
        prix_act := cl.Prix_A_Payer ;
        somme_a_payer := sp.Tarif * p_nbP ;
        Update Client Set Prix_A_Payer = somme_a_payer + prix_act
        Where id = p_idCl ;
        Commit ;
    Else
        Rollback ;
    End If ;
End ;
/
```

2. Soit le schéma de base de données suivant :

Classe (Filiere, Niveau, Effectif)

Matiere (Nom, Type, Vol_Horaire, Coefficient)

Etat (Filiere, Niveau, Matiere, DureeEffec, Etat)

Ecrivez une transaction qui à chaque fois qu'une classe suit une matière, incrémente la durée effectuée et modifie si nécessaire l'état (En attente, Encours, Fini)