Architecture orientée service
Bases de données avancées
partie I : Transactions

Jean-Marie Pécatte
jean-marie.pecatte@iut-tlse3.fr

SGBD
Gestion des transactions

Gestion des transactions

Définition

Une transaction est une unité atomique d'interaction composée de plusieurs commandes SQL qui préserve la cohérence de la base de données

➔Soit toutes les commandes sont exécutées

➔Soit aucune n'est exécutée

Exemple : *virement bancaire d'une somme de 1000 € d'un compte A vers un compte B*

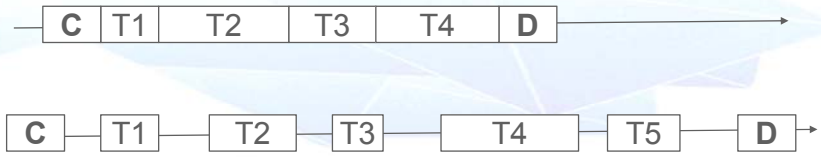
UPDATE compte SET montant = montant –1000 WHERE N°Compte = 'A'

UPDATE compte SET montant = montant +1000 WHERE N°Compte = 'B'

Pour que le virement soit correct il faut être sur que les 2 UPDATE soient exécutés

Notion de session

- Une session est la période qui s'écoule entre la connexion d'un utilisateur ou d'un programme à une base de données et la déconnexion
- Pendant une session, plusieurs transactions sont exécutées en séquence de manière contigüe ou pas (cas d'une session interactive)



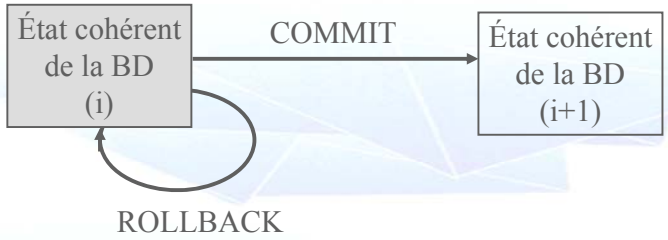
Gestion des transactions

Une transaction vérifie les propriétés ACID :

- A (Atomicité) correspond à "tout" ou "rien"
- C (Cohérence) passage état cohérent à état cohérent
- I (Isolation) entre les effets d'une transaction sur une autre tant qu'elle n'est pas validée
- D (Durabilité) persistance des m-à-j d'une transaction validée

Gestion des transactions

Une transaction se termine
soit par la commande COMMIT (validation) → "tout"
soit par la commande ROLLBACK (annulation) → "rien"



Gestion des transactions

En SQL2, il n'y a pas de commande pour débuter une transaction. Le début de la session (ou la fin de la transaction précédente) est considéré comme le début de la transaction.

Certains SGBD fonctionnent en mode AUTOCOMMIT, c'est-à-dire que chaque commande est validée automatiquement (pas de rollback possible)

Il est alors nécessaire d'introduire une commande de début de transaction à savoir BEGIN TRANSACTION.

Gestion des transactions

Points de sauvegarde intermédiaires
il est possible d'introduire des points de sauvegarde intermédiaire à l'intérieur d'une transaction
Création d'un point de sauvegarde
SAVEPOINT un
Annuler les modifications jusqu'à un point de sauvegarde
ROLLBACK TO un
Supprimer un point de sauvegarde
RELEASE SAVEPOINT un

Gestion des transactions

Concurrence :

une transaction conserve la cohérence de la BD, une succession de transactions aussi.

mais un SGBD est un système de gestion de données partagées et il doit permettre l'accès simultané aux mêmes données → exécution en parallèle de plusieurs transactions

→ Apparition de nombreux problèmes lors des accès concurrents :

- lecture impropre, lecture non reproductible, lecture fantôme, …
- apparitions d'inconsistances

Gestion des transactions

- Exemple :

M. Dupont arrive à la gare à 16h avec une réservation pour Paris dont le départ est prévue à 20h. Il souhaite échanger sa réservation pour un train partant plus tôt.

Le guichetier lui indique qu'il reste une seule place dans le train de 17h03 ; M. Dupont accepte.

Le guichetier annule la réservation pour 20h et demande une réservation pour 17h03.

Entre temps, la place disponible pour 17h03 a été attribuée par un autre guichetier.

Le guichetier décide donc de reprendre la réservation pour 20h.

Mais entre temps, cette réservation a été délivrée à un autre client.

M. Dupont ne peut plus regagner Paris !!!

Gestion des transactions

M. Dupont trouve une place à 17h03
SELECT nbpl FROM Voyage
WHERE dest='Paris' and depart='17:03'
Résultat : 1

M. Dupont libère sa place pour 20h
UPDATE Voyage SET nbpl = nbpl +1
WHERE dest='Paris' and depart='20:00'
Résultat : 1

M. Dupont veut réserver la place à 17h03
UPDATE Voyage SET nbpl = nbpl - 1
WHERE dest='Paris' and depart='17:03'
Refusé ! Plus de place !

M. Dupont veut récupérer sa place pour 20h
UPDATE Voyage SET nbpl = nbpl - 1
WHERE dest='Paris' and depart='20:00'
Refusé ! Plus de place !

M. X réserve la place pour 17h03
UPDATE Voyage SET nbpl = nbpl - 1
WHERE dest='Paris' and départ='17:03'
Résultat : 0

M. Y réserve la place qui vient de se libérer pour 20h
UPDATE Voyage SET nbpl = nbpl - 1
WHERE dest='Paris' and départ='20:00'
Résultat : 0

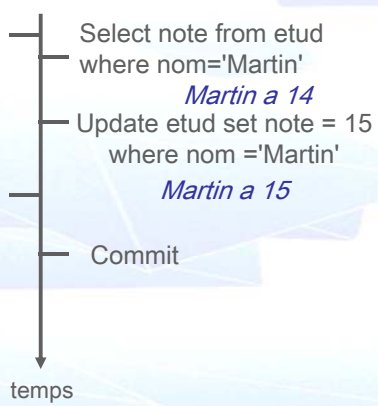
temps

Gestion des transactions : Isolation

- Le programmeur regroupe ses commandes SQL dans des transactions
- Plusieurs programmes, donc transactions, peuvent s'exécuter simultanément
  - → quelle influence peut avoir une transaction sur le résultat d'une autre ?
- En fonction du niveau d'Isolation entre 2 transactions, plusieurs problèmes sont possibles dues aux effets d'une transaction sur une autre
  - Lecture impropre
  - Lecture non reproductible
  - Lecture fantôme

Gestion des transactions : Isolation

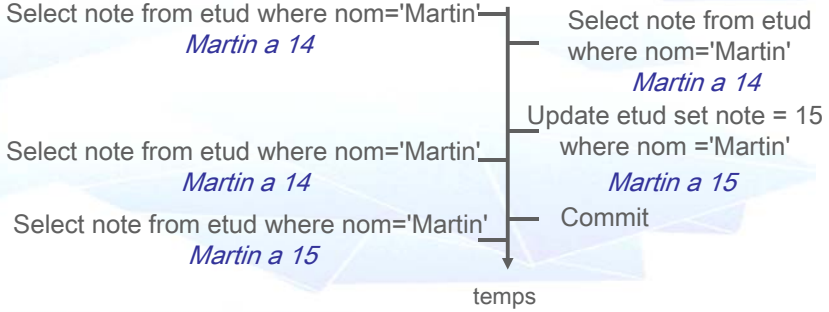
Lecture impropre : une transaction lit des données écrites par une transaction non encore terminée.



La note de Martin vient de changer alors que la modification n'a pas été validée par (2)

Gestion des transactions : Isolation

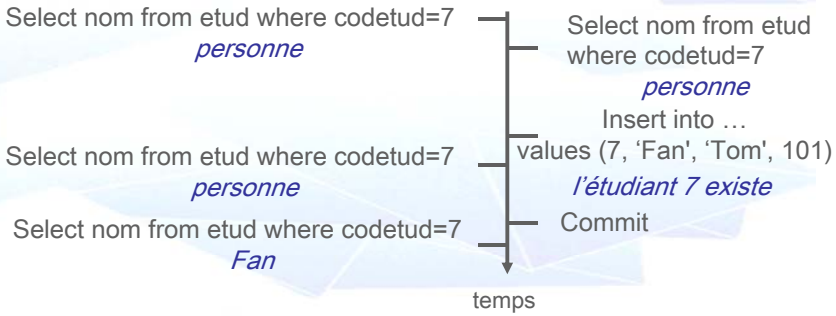
Lecture non reproductible : une transaction relit des données qu'elle a précédemment lues et trouve qu'elles ont été modifiées par une autre transaction.



La note de Martin vient de changer alors (1) n'a apporté aucune modification

Gestion des transactions : Isolation

Lecture fantôme : une transaction relit des données vérifiant une certaine condition et trouve que des tuples supplémentaires satisfaisant la condition ont été insérés par une autre transaction.



La transaction (1) voit apparaître un nouvel étudiant de code 7

Gestion des transactions : Isolation

Le standard SQL2 définit 4 niveaux d'isolation des transactions.
Ces niveaux sont définis en fonction des problèmes que l'on peut rencontrer entre des transactions concurrentes.

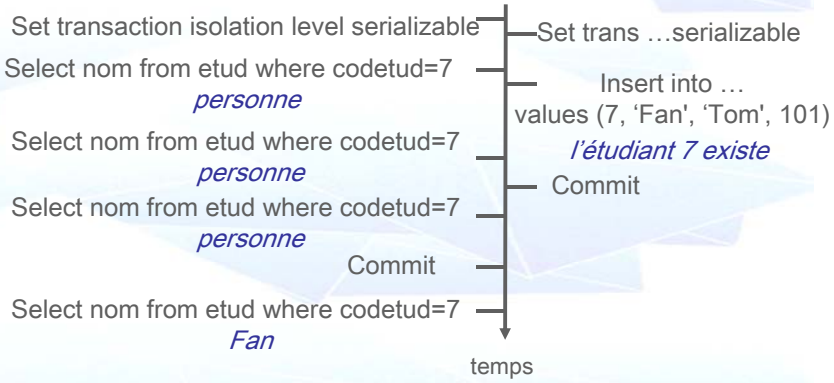
Niveau d'isolation	Lecture Impropre	Lecture non reproductible	Lecture fantôme
Read uncommited	Possible	Possible	Possible
Read committed	Impossible	Possible	Possible
Repeatable read	Impossible	Impossible	Possible
Serializable	Impossible	Impossible	Impossible

Gestion des transactions : Isolation

- Le programmeur peut donc choisir le niveau d'isolation d'une transaction ; pour cela il faut utiliser la commande SQL : **SET TRANSACTION ISOLATION LEVEL**
  { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }
- Cette commande doit être la première d'une transaction
- Si la commande est absente, c'est le mode d'isolation par défaut qui est appliqué (dépend de la configuration du SGBD)
- Isolation en entrée seulement, pas en sortie :
  - Chaque transaction, via le mode d'isolation, peut choisir ce qu'elle veut voir ou pas des autres transactions.
  - Par contre, une transaction ne peut pas choisir ce qu'elle rend visible ou pas de ce qu'elle fait.

Gestion des transactions : Isolation

Résolution du pb de la lecture fantôme : si on place la transaction (1) en mode sérialisable, il n'y a plus de problème.



La transaction (1) ne voit pas apparaître le nouvel étudiant

SGBD
Concurrence

Gestion des transactions : concurrence

Autre problème lors de l'exécution en concurrence de transactions : comment gérer l'accès simultané aux mêmes données ?

Deux transactions peuvent-elles modifier les mêmes données au même instant ? ➔ NON !

Solutions :

- Deux techniques :
  prévention des conflits : contrôle de concurrence pessimiste
  détection des conflits : contrôle de concurrence optimiste

Gestion des transactions : concurrence

Contrôle de concurrence pessimiste

- Utilisation de verrous (lecture ou écriture)
  - Un verrou 'écriture' n'autorise aucun autre verrou en 'écriture'
- Un verrou est posé sur un granule (une table, une page, des tuples, …)
- Le choix d'une unité de verrouillage fine minimise les risques de conflits mais maximise la complexité et le coût du verrouillage
- Verrouillage à deux phases
  - Phase d'acquisition des verrous (à n'importe quel moment de la transaction)
  - Phase de libération des verrous (une seule fois, à la fin de la transaction)

Gestion des transactions : concurrence

- Le verrouillage peut être explicite : commande LOCK
  LOCK TABLE <nom> IN <mode> MODE

➔ Nombreux modes de verrouillage

mode lignes partagés (Row share)
mode lignes exclusives (Row exclusive)
mode table partagée (Share)
mode partage exclusif de lignes (Share row exclusive)
mode table exclusive (Exclusive)
mode accès exclusif (Access exclusive)

Exemple :

Pose d'un verrou en accès exclusif sur la relation Sportif : l'accès à la relation sportif est impossible à toute autre transaction

LOCK TABLE Sportif IN ACCESS EXCLUSIVE MODE

Gestion des transactions : concurrence

- Le verrouillage peut être implicite : le verrou est posé automatiquement par la commande UPDATE, DELETE, INSERT ; il s'agit alors d'un verrou en mode ligne exclusif (seuls les tuples concernés par la modification sont verrouillés)
- Verrou mode ligne partagée

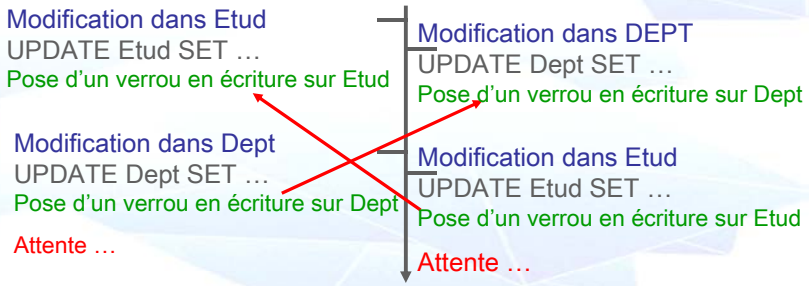
Les tuples verrouillés ne peuvent être ni modifiés ni supprimés par une autre transaction.

Ce verrou est posé de manière implicite par la commande : SELECT … FOR UPDATE (verrouillage pour modification)

Gestion des transactions : concurrence

Problèmes liés à l'utilisation de verrous

- Interblocage entre 2 transactions



Solution : algorithme de détection de boucles dans le graphe d'attente ; le système choisit alors d'arrêter une transaction (Rollback).

SGBD Oracle

Transactions

- Pas de commande de début de transaction
- 3 modes d'isolation
  - READ COMMITTED
  - SERIALIZABLE
  - READ ONLY (comme serializable mais en lecture seule)
- Points de sauvegarde intermédiaire
  - SAVEPOINT …
  - ROLLBACK TO SAVEPOINT …