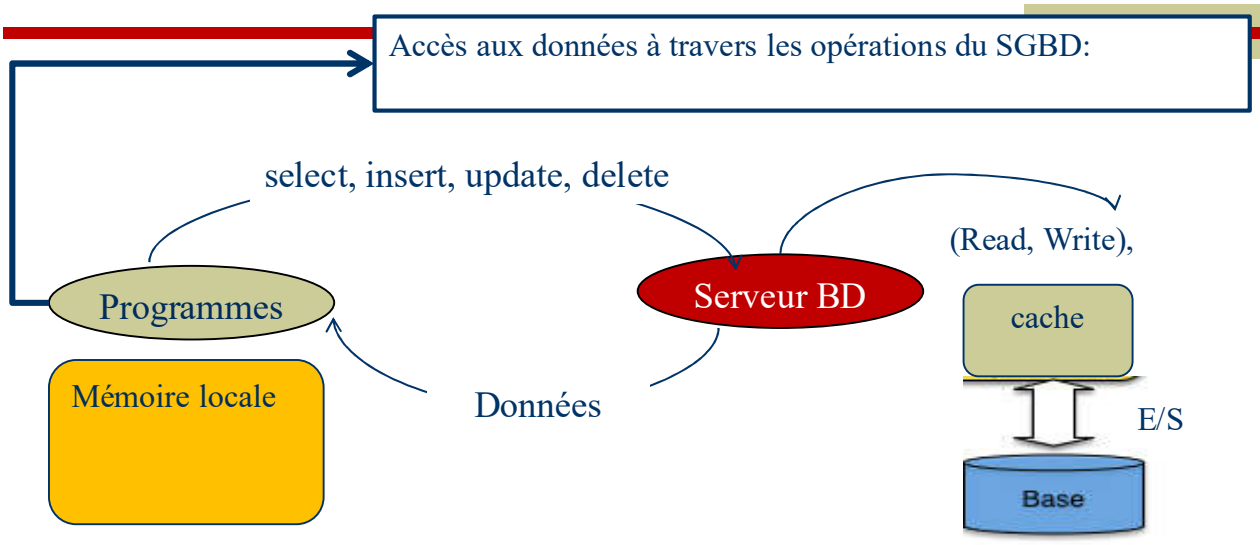
ENICARTHAGE
المدرسة الوطنية للمهندسين بقرطاج
Ecole Nationale d'Ingénieurs de Carthage

Gestion des transactions

Introduction



•Une transaction s'exécute dans le cadre d'un **processus client** connecté au **processus serveur** (SGBD).
•Plusieurs transactions peuvent s'exécuter l'une suite de l'autre dans un même processus : elles sont dites **sérielles** (si pas de parallélisme).
•En revanche, deux processus **distincts** engendrent des **transactions concurrentes**.

Exemples de transactions

- Transactions bancaires,
- Commandes dans un magasin de vente par correspondance,
- Réservation de places pour les spectacles
- Réservations de places sur des vols,
- Inscriptions des étudiants, Etc,…

Plan

- I- Introduction aux transactions
  - 1. Notion de transaction
  - 2. Propriétés ACID
  - 3. Points de repère
  - 4. Etats de transactions
- II- Théorie de la concurrence
  - 1.Introduction: la concurrence de transactions
  - 2. Problèmes des accès concurrents:
  - 3. Sérialisation
  - 4.Niveaux d'isolation sous SQL
  - 5. Récupérabilité & Reprise sur panne
  - 6. Techniques de contrôle de concurrence
    - Verrouillage 2 phases,
    - Estampillage

1.Notion de transaction

- Définition 1:

> Une transaction est un ensemble d'ordres SQL (lectures/écritures) qui constitue une unité logique de traitement qui fait passer la base d'un état cohérent à un état cohérent.

- On représente une transaction par une séquence de lecture et d'écriture parce que **Les opérations effectuées par un processus client sont inconnues du serveur de données**

1.Notion de transaction

Début , fin d'une transaction

- Il n'existe pas d'ordre PL/SQL ou SQL qui marque le début d'une transaction: En SQL, le début de transaction est implicite.
  - le BEGIN d'un programme PL/SQL n'est pas forcément synonyme de son commencement.
- Elle se termine implicitement :
  - à la première commande SQL du LDD ou du LCD rencontrée ;
  - à la fin normale d'une session utilisateur avec déconnexion ;
  - à la fin anormale d'une session utilisateur (sans déconnexion).
- Une transaction se termine explicitement par les instructions SQL de validation: COMMIT ou d'annulation:ROLLBACK. (ou abort)
- COMMIT:
  - la transaction s'est bien terminée,
  - les modifications qu'elle a exécutées sont rendues visibles aux autres transactions.
  - Enregistrement physique

Notations

- Transaction Ti: Séquence d'opérations terminée par Commit/Rollback
- L'indice *i* identifie la transaction et ses opérations
- Opérations: lire, écrire, commit, rollback
  - Ri[*x*]: lecture de la donnée *x* dans la transaction Ti
  - Wi[*x*]: écriture de la donnée *x* dans la transaction Ti
  - ci : validation (commit) de la transaction Ti
  - ai: annulation (rollback) de la transaction Ti

Exemple

Alice souhaite verser à Bob 100 euros.
• Le SGBD doit effectuer 2 opérations sur la table COMPTE :
– Retirer 100 euros du compte d'Alice
– Ajouter 100 euros au compte de Bob

Begin transaction (optionnel)

t1 **Select solde from compte where NC=0;**

t2 **Update COMPTE SET SOLDE=SOLDE-100 WHERE NC=0;**

t3 **Select solde from compte where NC=1;**

t4 **Update COMPTE SET SOLDE=SOLDE+100 WHERE NC=1;**

t5 **Commit/rollback**

COMPTE

NC	CLIENT	SOLDE
0	Alice	1000
1	Bob	750
...	...	...

COMPTE

NC	CLIENT	SOLDE
0	Alice	900
1	Bob	850
...	...	...

Somme : 1750

T1: R1[x]W1[x]R1[y]W1[y]C1: en cas de validation

2.Propriétés ACID

- Définition 2

Une transaction est composée d'une suite de requêtes dépendantes à la base qui doivent vérifier les propriétés d'**atomicité, de cohérence, d'isolation et de durabilité, résumées par le vocable ACID [Harder&Reuter, ACM CS 15(4), 1983]**

On parle d'ACIDité ou de propriétés ACID d'une transaction

Garantie de la commande commit

Propriétés ACID: Exemple

Exemple:
Une transaction qui transfert 100 Eur du compte A vers le compte B

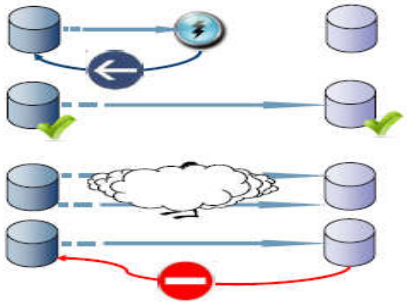
1. Lire(A)
2. A := A − 100
3. Ecrire(A)
4. Lire(B)
5. B := B +100
6. Ecrire(B)

T2: afficher somme A+B

- **Cohérence**:
  - La base est cohérente si la somme A+B ne change pas suite à l'exécution de la transaction .
- **Atomicité**: (Garantie du Rollback)
  - Si la transaction "´echoue" après l'´etape 3, alors, le système doit s'assurer que les modifications de A ne soient pas persistantes.

- **Durabilité** (Garantie du commit)
  - Une fois l'utilisateur est informé que la transaction est validée(commit;), il n'a plus à s'inquiéter du sort de son transfert
- **Isolation**:
  - Si entre les étapes 3 et 6, une autre transaction est autorisée à accéder à la base, alors elle "verra" un état incohérent (A+B) est inférieur à ce qu'elle doit être). L'isolation n'est pas assurée

A:
C:
I :
D:



2.Propriétés ACID

- **A**tomicité
  - Soit toutes les modifications effectuées par une transaction sont enregistrées dans la BD, soit aucune ne l'est.
- **C**ohérence (Consistency)
  - Une transaction fait passer une BD d'un état cohérent à un autre état cohérent. **Un état cohérent est un état dans lequel les contraintes d'intégrité sont vérifiées.** ➔ Différer la vérification de certaines contraintes ?
- **I**solation
  - Une transaction se déroule sans être perturbée par les transactions concurrentes : tout se passe comme si elle se déroulait seule.
  - Même si plusieurs transactions peuvent être exécutées en concurrence, aucune n'est censée prendre en compte les autres transactions ;
    - i.e pour chaque paire Ti, Tj, pour Ti tout se passe comme si Tj s'est terminée avant le début de Ti ou bien qu'elle commence après la fin de Ti (les résultats intermédiaires de Tj lui sont invisibles)
    - Exp: les lignes de comptes pour les deux utilisateurs doivent être protégées.
- **D**urabilité
  - Une fois qu'une transaction a été confirmée (la primitive se termine normalement), le SGBD garantit qu'aucune modification qu'elle a effectuée ne sera perdue même en cas d' : interruption, pannes du système d'exploitation, « crash » de disque, etc.

3.Points de repères

- Possible de subdiviser une transaction en plusieurs étapes
  - en sauvant les informations modifiées à la fin de chaque étape,
  - en gardant la possibilité soit
    - de valider l'ensemble des mises à jour,
    - d'annuler une partie des mises à jour à la fin de la transaction.

  → Insérer des points de repère, ou SAVEPOINT.
    - La création d'un point de repère se fait avec l'instruction : PL/SQL SAVEPOINT pointRepere;
    - Pour annuler la partie de la transaction depuis un point de repère : ROLLBACK TO [SAVEPOINT] pointRepere;

3.Points de repères

- Exemple:

Code PL/SQL	Commentaires
BEGIN INSERT INTO Compagnie VALUES ('C2',2, 'Place Brassens', 'Blagnac', 'Easy Jet');	Première partie de la transaction.
SAVEPOINT P1; UPDATE Compagnie SET nrue = 125 WHERE comp = 'AF'; UPDATE Compagnie SET ville = 'Castanet' WHERE comp = 'C1';	Deuxième partie de la transaction.
SAVEPOINT P2; DELETE FROM Compagnie WHERE comp = 'C1';	Troisième partie de la transaction.
-- ROLLBACK TO P1;	Première partie à valider.
-- ROLLBACK TO P2;	Deuxième partie à valider.
-- ROLLBACK TO P3	Troisième partie à valider.
ROLLBACK;	Tout à invalider.
COMMIT; END;	Valide la ou les sous-parties.

SAVEPOINT P3;

4.Etats d'une transaction

Naissance de la transaction

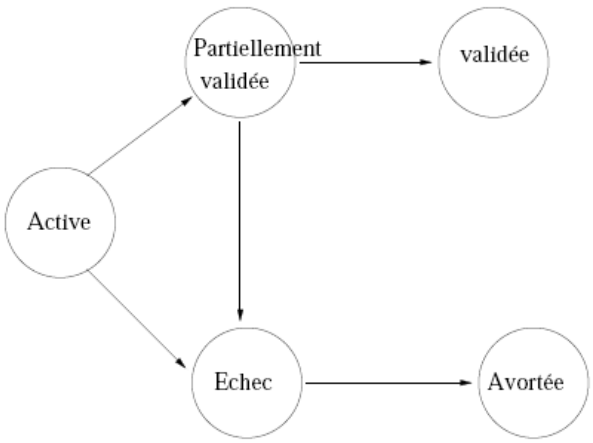
• le SGBD lui associe un identificateur unique + d'autres informations

1. **Active** : la transaction reste dans cet état durant son exécution: :
   - consommation de ressource (verrouillages, journalisation, …)
   - choisir le niveau d 'isolation approprié
2. **Partiellement validée** : juste après l'exécution de la dernière opération
3. **Echouée** : après avoir découvert qu'une exécution "normale" ne peut pas avoir lieu.
4. **Avortée (abotred)** : Après que toutes les modifications faites par la transaction soient annulées (Roll back).
   - Deux options
     - Ré-exécuter la transaction
     - Tuer la transaction
5. **Validée** : après l'exécution avec succès de la dernière opération

   état *confirmé* ***(COMMITTED)***

   libération de ressources

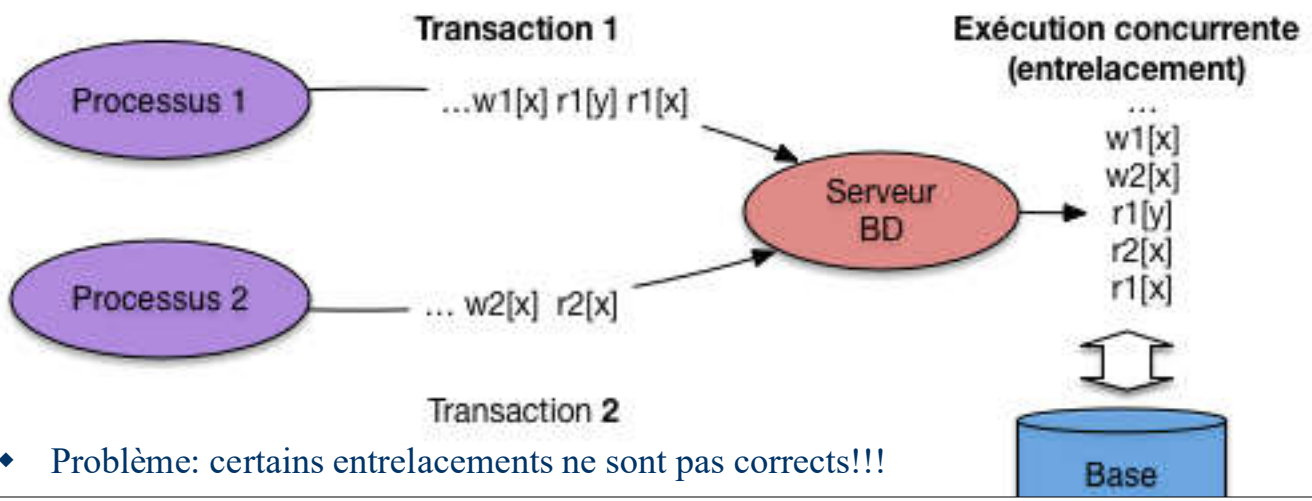
   ➔ confirmer le plus tôt possible pour éviter de monopoliser les ressources



II-Théorie de la concurrence

1.Concurrence de transactions

- Concurrence = entrelacement des opérations des transactions
- Vision client-serveur
  - Les transactions = Processus clients qui envoient des demandes au SGBD (demandes de lecture ou d'écriture de tuples)
  - Le SGBD = serveur qui reçoit ces demandes et les exécute
  - Le SGBD exécute les opérations en séquence, pas en parallèle!



- Problème: certains entrelacements ne sont pas corrects!!!

Vocabulaire

Base
↓
Table
↓
Page
↓
Tuple

- Il y a un accès concurrent lorsque plusieurs utilisateurs (transactions) accèdent en même temps à la même donnée (même granule de concurrence) dans une base de données.
- **Granule de concurrence**: unité de données dont les accès sont contrôlés individuellement par le SGBD.
- Dans un SGBD cela peut être un tuple, une page ou bien une table.
- Concurrence parfaite: toute opération est exécutée dès son arrivée
- **Une opération** est un accès élémentaire à un granule(lecture/écriture).
- **Exécution:** Séquence d'opérations (lectures /écritures) obtenues en entrelaçant les diverses opérations des différentes transactions tout en respectant l'ordre interne des opérations de chaque transaction
  - Une exécution est dite aussi histoire ou o**rdonnancement (Schedule)**

Problèmes

- En BD de multiples utilisateurs doivent pouvoir accéder à la base de donnée en même temps: il faut s'assurer que les actions des uns ne seront pas préjudiciables aux autres.
- **Problèmes d'accès concurrents** Peuvent être classés en deux catégories:
  - Défauts d'isolation: menant à des incohérences: défauts de sérialisabilité,
  - Difficultés dus à une mauvaise prise en compte des commit et rollback, ou défauts de recouvrabilité (récupérabilité)
- Les exemples qui suivent ne couvrent pas l'exhaustivité des situations d'anomalies, mais illustrent les principaux types de problèmes.

II-THÉORIE DE LA CONCURRENCE

2- Problèmes des accès concurrents

2.Problèmes des accès concurrents

- Problèmes posés par les accès concurrents
  - Perte de mise à jour (lost update) –la plus difficile
  - Lecture impropre (lecture de valeurs non validées)
  - Introduction d'incohérence
  - Lecture non reproductible (non répétable)
  - Lecture fantôme

Exemple

- 1 table : EMP (NE, Nom, Sal)
- 3 tuples :

NE	Nom	sal
1	Ib-Khalil	2000
2	Eya	2100
3	Aicha	1600

… et deux utilisateurs :

Omar et Kenza qui feront des accès concurrents à la base

2.1 Perte de mise jour (Lost update) W-W

- T1: Omar met à jour le salaire de Ib-khalil en l'augmentant de 50d
- T2: Kenza augmente également le salaire de Ib-khalil de 1000d
- La séquence des opérations reçue est la suivante:

Temps	T1 (Omar)	T2(Kenza)	X:salaire (BD)
T1			2000
t2		Read (X)	
t3	Read(X)	X=x+1000	
t4	X= X+50	Write (x)	
t5	Write (x)		

- On obtient une augmentation du salaire désigné de 50 au lieu de 1050!!!
- Le problème est clairement issu d'une mauvaise imbrication des opérations de T1 et T2: **T1 lit et modifie une information que T2 a déjà lue en vue de la modifier. (écriture simultannée)**

➔ SI T1 MODIFIE UN N-UPLET PUIS T2 LE MODIFIE AUSSI ALORS T1 PERD SA MODIFICATION EN COURS DE TRANSACTION ET VICE VERSA.

2.2 Lecture impropre (dirty read): a. Donnée non confirmée (W-R)

- Une lecture impropre (Dirty Read) ou modification temporaire a lieu quand Ti lit un objet A modifié par Tj alors que Tj n'a pas encore été validé
- Exemple1:

T_1	T_2	BD
		A = 50
	A = 70	
	write A	A = 70
read A (70 est lu)		
	rollback (La valeur initiale de A est restaurée)	A = 50

- T1 utilise la valeur A=70 écrite par T2, qui est finalement annulée
- Non respect de l'isolation: une transaction est autorisée à voir les résultats d'une autre transaction active (non encore confirmée)
- L'annulation de T2 implique l'annulation de T2 aussi (annulation en cascade)
- Durabilité non respectée: T1, qui est validée, doit être annulée

2.3. Incohérence

- Une incohérence apparaît lorsque des données liées par une contrainte d'intégrité sont mises à jour par deux transactions dans des ordres différents, de sorte à enfreindre la contrainte
- Exemple, soient deux données A et B devant rester égales.

T1	T2	**BD** CI (A=B) 10	
		A	B
A=A+1		11	10
	A=A*2	22	10
	B=B*2	22	20
B=B+1		22	21

- L'exécution de la séquence d'opérations
  - {T1 : A = A+1 ; T2 : A = A*2; T2 : B = B*2 ; T1: B=B+1}
- Rend en général A différent de B,
  - du fait de la non-commutativité de l'addition et de la multiplication

2.4.Lecture non reproductible (unrepeatable read)

- T1 et T2 transactions concurrentes
- T1 effectue plusieurs fois la lecture du même tuple

T1(Omar)	T2(Kenza)	**BD** sal
SELECT E.SAL FROM EMP E WHERE E.NOM = 'Ib.khalil'		2000
	UPDATE EMP SET SAL = 2050 WHERE NOM = 'Ib.Khalil'	2050
SELECT E.SAL FROM EMP E WHERE E.NOM = 'Ib.khalil'		2050

temps

- En principe, T1 doit obtenir à chaque lecture la même valeur
- Mais, si entre 2 lectures de l'objet, celui-ci est modifié par T2, T1 n'obtient plus le même valeur : lecture non reproductible !
  - Le problème ne survient pas si les mises à jour sont isolées, c'est-à-dire non visibles par une autre transaction avant la fin de la transaction(T1).

2.5 Lecture fantôme (phantom read)

T1(Omar)	T2(Kenza)	**AVG(E.SAL)**
SELECT AVG(E.SAL) FROM EMP E		1900
	INSERT INTO EMP VALUES (4, 'Yazid', 2300) , commit	
SELECT AVG(E.SAL) FROM EMP E		2000

- T1 ré-exécute une requête renvoyant la moyenne des salaires et trouve que sa valeur a changé à cause d'une transaction récemment validée.

NE	Nom	sal
1	Ib.Khalil	2000
2	Eya	2100
3	Aicha	1600
4	Yazid	2300

T_1 n'a pas vu l'ajout de 4 dans Emp par T_2, pour T_1 4 est un objet fantôme

Un tuple peut être utilisé pour produire le résultat de la requête (implicite) e.g. requêtes d'agrégats. Une modification de l'un des tuple utilisé pour calculer la requête causera donc un problème.

2.6. Conclusion

- Les problèmes qui sont survenus sont dus à un accès concurrent des opérations successives
  - Cet accès n'a pas su isolé les différentes transactions actives
- L'objectif de CC est de:
  - dérouler les opérations des transactions concurrentes en **isolation totale:** ➔ la Sérialisation
- La **sérialisabilité** permet de répondre au problème de **l'isolation** dans le cas de transactions concurrentes