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

Gestion des transactions

Plan

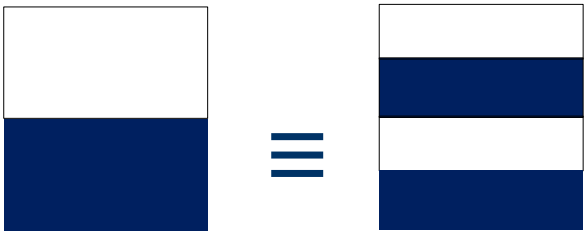
- I- Introduction aux transactions
  - 1. Notion de transaction
  - 2. Propriétés ACID
  - 3. Points de repère
  - 4. Etats de transactions
- II- Théorie de la concurrence
  - 1.Introduction: la concurrence de transactions
  - 2. Problèmes des accès concurrents:
  - 3. Théorie de Sérialisation
  - 4.Niveaux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

II-THÉORIE DE LA CONCURRENCE

3- La sérialisation

3.1 Objectif de la sérialisabilité

- Modifier l'entrelacement pour produire une exécution correcte des transactions
- Trouver un ordre **non sériel** des opérations des différentes transactions qui **produit le même résultat qu'un ordonnancement sériel(séquentiel)**
- **Un tel ordonnancement est dit sérialisable**
  - Retardement des opérations,
  - Perte de performances



3.4 Permutations / conflits d'opérations:

- L'ordre des opérations de Lectures (R) et écritures (W) est important dans la sérialisabilité
  - T1 lit et T2 lit => l'ordre n'est pas important (pas de conflit)
  - Si T1 et T2 lisent ou écrivent des items non communs (Granules) => pas de conflits
  - Si l'une écrit un granule qui est lu ou écrit par la suite par une autre transaction => l'ordre d'exécution est fondamental !
- Définitions:
  - Opérations conflictuelles:
    - Deux opérations pi[x] et qj[y] sont en conflit si x = y, i ≠ j, p ou q est une écriture.➔deux transactions accèdent au même nuplet, et (au moins) une veut le modifier.
    - *pi*[*x*] et *qj*[*y*] ne sont pas permutables
  - Opérations permutables:
    - Deux opérations A et B sont dites permutables: ssi l'exécution A;B (A suivi de B) donne le même résultat que B;A

3.5 Équivalence des exécutions

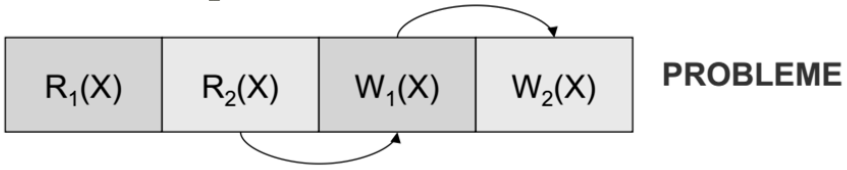
- Une exécution concurrente est sérialisable si elle peut être transformée en une exécution en série **équivalente** par une suite de permutations d'opérations non conflictuelles
- Deux exécutions d'un même ensemble de transactions sont équivalentes ssi :
  - Elles sont constituées des mêmes opérations,
  - Elles produisent le même état final de la BD et les mêmes résultats pour les transactions :
    - Les lectures produisent les mêmes résultats.
    - Les écritures sont réalisées dans le même ordre.

Exemple d'équivalence d'exécutions

- Transactions:
  - **T1**: R1[*x*] W1[*y*] W1[*x*] c1
  - **T2**: W2[*x*] R2[*y*] W2[*y*] c2
- Exécutions
  - → **H1**: W2[*x*] R2[*y*] R1[*x*] W2[*y*] W1[*y*] c2 W1[*x*] c1
  - → Dans H1: R1[*x*] et W2[*y*] sont permutables car elles travaillent sur des granules différents
  - → H1.1: W2[*x*] R2[*y*] W2[*y*] c2 R1[*x*] W1[*y*] W1[*x*] c1
    - → H1 est équivalente à H1.1 (T2,T1)
    - → H1 s'est transformé en ordonnancement sériel par permutations des opérations non conflictuelles
    - → H1 est sérialisable

3.6 Test de sérialisabilité

- Idée: Un ordonnancement est sérialisable si toutes les opérations conflictuelles sont effectuées dans le même ordre que dans un ordonnancement séquentiel→ la notion de **précédence** devient fondamentale



- La précédence: propriété indiquant qu'une transaction a accompli une opération Oi sur une donnée avant qu'une autre transaction ne réalise une opération Oj;
- Oi et Oj n'étant pas permutables ({Oi;Oj}≠{Oj;Oi}) (opérations conflictuelles).
- La relation de précédence entre transactions peut être représentée par un graphe : Graphe de précédence (Precendency graph)

3.7 Graphe de précédence

- ou de sérialisation: Graphe dont les nœuds représentent les transactions et dans lequel il existe un arc de Ti vers Tj si Ti précède Tj dans l'exécution analysée.
- Protocole
  - Si G ne contient pas de cycles (boucles) alors l'ordonnancement est sérialisable (l'inverse n'est pas vrai)

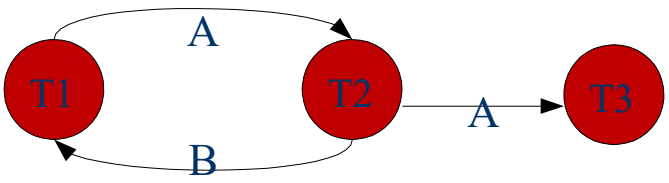
➔ L'ordre séquentiel équivalent correspond à un chemin dans le graphe

Exemple

O1
{T1:Lire A;
T2: Ecrire A;
T2: Lire B;
T3: Lire A;
T1: Ecrire B}

O1: R1[A]W2[A]R2[B]R2[A]R3[A]W1[B]

Opérations conflictuelles (non permutables):
- Sur A:
  - R1[A]-W2[A],
  - W2[A]-R3[A]
  - => précédence: T1→T2, T2→T3
- Sur B:
  - R2[B]-W1[B]
  - Précédence: T2→T1



Le graphe de précédence de O1 présente un cycle ➔ O1 n'est pas sérialisable par permutations

Cas particulier: Ordonnancement avec cycle mais sérialisable !

Temps	Transaction T_1	Transaction T_2	État de la BD A
t_1	**DébutTransaction**		1
t_2	**Lire**(A, a)		1
t_3	$a := a^2$		
t_4		**DébutTransaction**	
t_5		**Lire**(A, a)	1
t_6	**Écrire**(a, A)		1
t_7		$a := a + 10$	
t_8		**Écrire**(a, A)	11
t_9	**ConfirmerTransaction**		
t_{10}		**ConfirmerTransaction**	

- Cette exécution: R1[A]R2[A]W1[A]W2[A]c1c2
  - T1→T2 : →R1[A] W1[A] c1R2[A]W2[A] c2→1, 1, 1,11
  - T2→T1:→ R2[a] W2[a] c2 R1[a]W1[a]c2→1, 11, 11,121

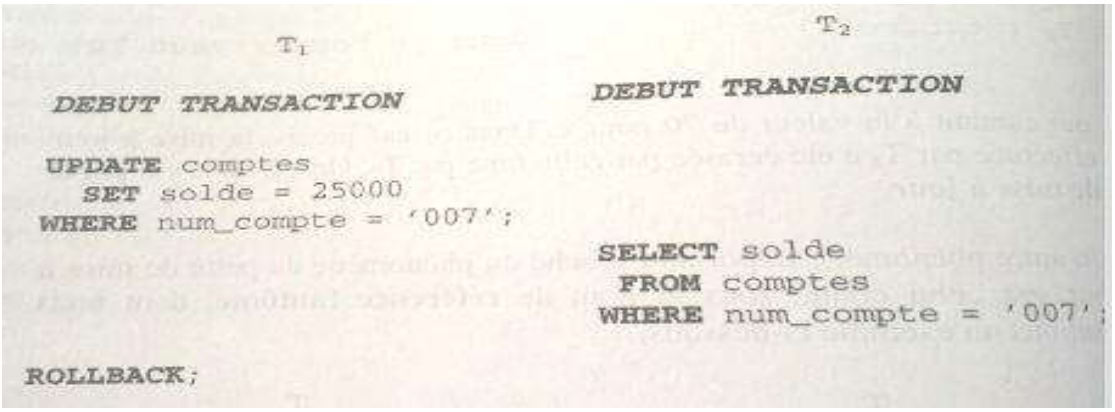
3.8. Résumé

- Un ordonnancement est sérialisable par permutation si les opérations conflictuelles sont effectuées dans le même ordre relatif à un ordonnancement séquentiel→ graphe de précédence sans cycle
- Si le graphe **ne** contient **pas** de cycle→ il est sérialisable par permutation →l'ordonnancement est sérialisable
- Un ordonnancement sérialisable, peut présenter un cycle sur le graphe de précédence
  → il n'est pas forcément sérialisable par permutation
- Un ordonnancement non sérialisable →n'est pas sérialisable par permutation
  - Le graphe de précédence contient forcément des cycles

Exercices

Exercice 1 (1/5)

- Préciser le problème pour chaque ordonnancement ci-dessous.
- **O1 :**



T_1	T_2
DEBUT TRANSACTION	DEBUT TRANSACTION
UPDATE comptes SET solde = 25000 WHERE num_compte = '007';	
	SELECT solde FROM comptes WHERE num_compte = '007';
ROLLBACK;	

Lecture impropre données non confirmée- modification temporaire

Exercice 1 (2/5)

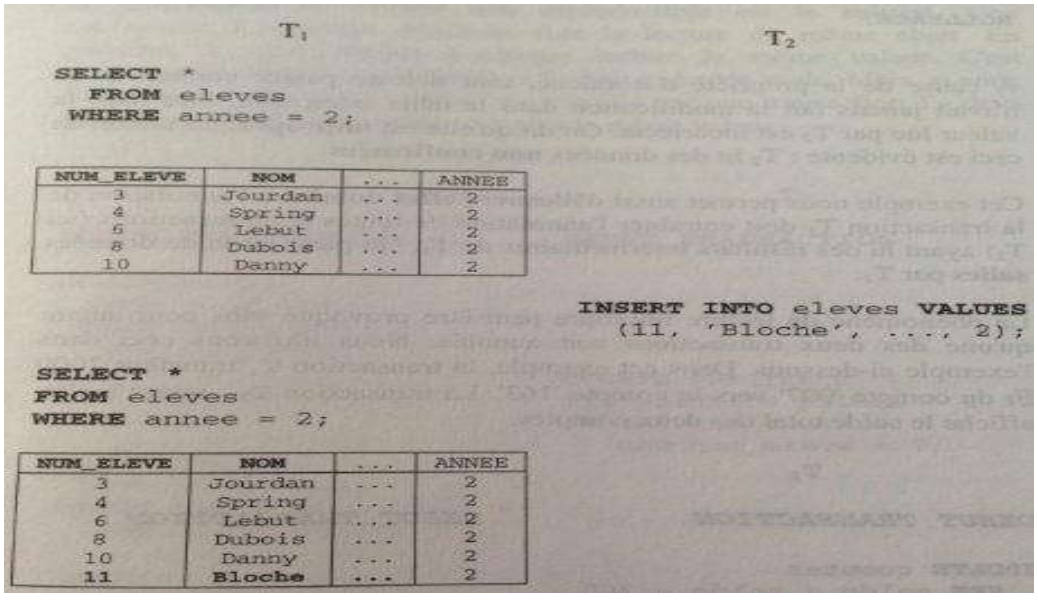
- O2:

T1	T2
SELECT points FROM Resultats WHERE num_cours = 5 AND num_eleve = 7 ;	
	UPDATE Resultats SET points = points * 1.1 WHERE num_cours = 5 AND nom_eleve = 7 ;
SELECT points FROM Resultats WHERE num_cours = 5 AND num_eleve = 7 ;	
COMMIT	

Lecture non reproductible

Exercice 1 (3/5)

- O3:



Lecture fantôme

Exercice 1(4/5)

- O4: Hypothèse: Ali et Fatma sont deux clients désirant réserver à travers deux transactions T1 et T2 respectivement 2 et 5 places dans le spectacle s où le nombre de places libres=50:

Temps	T1 (Ali)	T2(Fatma)	x:Nb_places de s dans spectacle=50
t1		Read (x)	50
t2	Read(x)		50
t3	X=x-2		50
t4	Write(x)	X=x-5	48
t5		Write (x)	45

- 45 places vides sur les 50 initiales alors que 7 places ont effectivement été réservées et payées!!!

Lost update

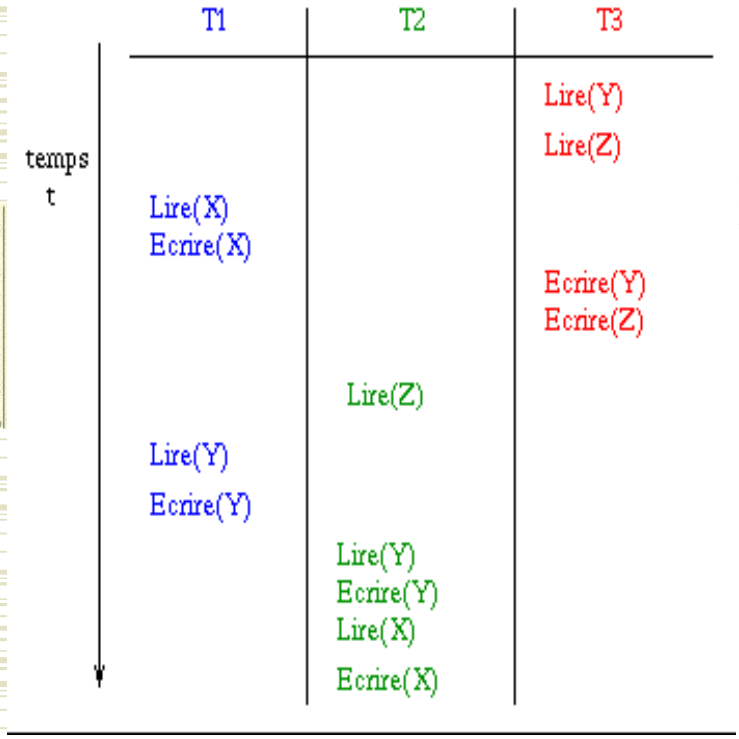
Exercice 1 (5/5)

- O5

Temps	T_5	T_6	$solde_x$	$solde_y$	$solde_z$	somme
t_1		début_transaction	100	50	25	
t_2	début_transaction	somme = 0	100	50	25	0
t_3	lire($solde_x$)	lire($solde_x$)	100	50	25	0
t_4	$solde_x = solde_x - 10$	somme = somme + $solde_x$	100	50	25	100
t_5	écrire($solde_x$)	lire($solde_y$)	90	50	25	100
t_6	lire($solde_z$)	somme = somme + $solde_y$	90	50	25	150
t_7	$solde_z = solde_z + 10$		90	50	25	150
t_8	écrire($solde_z$)		90	50	35	150
t_9	validation	lire($solde_z$)	90	50	35	150
t_{10}		somme = somme + $solde_z$	90	50	35	**175**
t_{11}		validation	90	50	35	**175**

- Incohérence

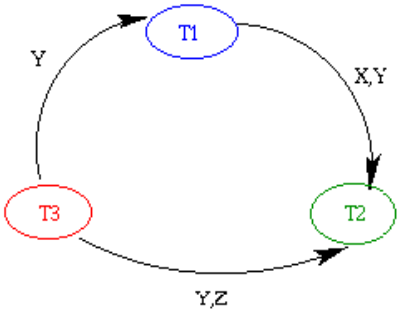
Exercice 2 (1/2)



Cette exécution peut être écrite de la façon suivante:
•R3[y]R3[z]R1[x]W1[x]W3[y] W3[z] R2[z] R1[y] W1[y] R2[y] W2[y] R2[x] W2[x]

Exercice 2 (2/2)

- R3[y]R3[z]R1[x]W1[x]W3[y] W3[z] R2[z] R1[y] W1[y] R2[y] W2[y] R2[x] W2[x]
- Les opérations conflictuelles (non permutables sur x):
  - R1[x]-W2[x], w1[x] R2[x], W1[X]-W2[X]: T1 →T2
- Les opérations conflictuelles sur y:
  - R3[y]-W1[y], R3[y]-W1[y],W3[y]-R1[y],W3-R1, W3-W1, W3-R2, W3-W2,
  - w1[y]-R2[y], R1[y]-W2[Y], w1[y]-W2[y]
  - T3→T1
  - T3→T2
  - T1→T2
- Les opérations conflictuelles sur z:
  - W3[z] R2[z]: T3→T2



II-THÉORIE DE LA CONCURRENCE

4- Niveaux d'isolation