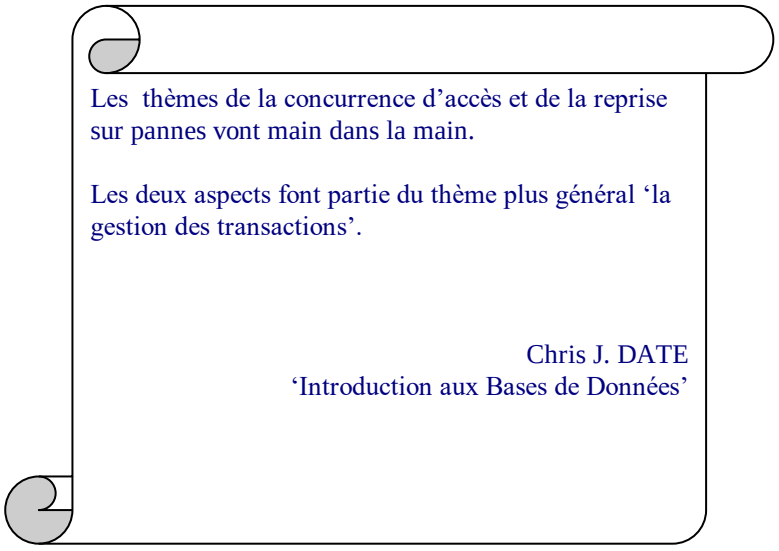
SGBDR ORACLE

La gestion des transactions et la concurrence d'accès

Appliquez les différents types de transactions sur la B.D. BALNEO en vous inspirant des exemples fournis. Ajoutez vos tests et constats dans le document que vous déposerez pour évaluation



Auteur : C. ROELS

Objectif : Connaître les principes de la gestion des transactions.
Comprendre les difficultés liés à cette gestion.

Contenu :

I. Trois problèmes liés à la concurrence d'accès.

II. Le verrouillage

III. Les 3 problèmes de concurrence revisités

IV. Le blocage

V. La sérialisabilité

VI. Le niveau d'isolation

I. Trois problèmes liés à la concurrence d'accès.

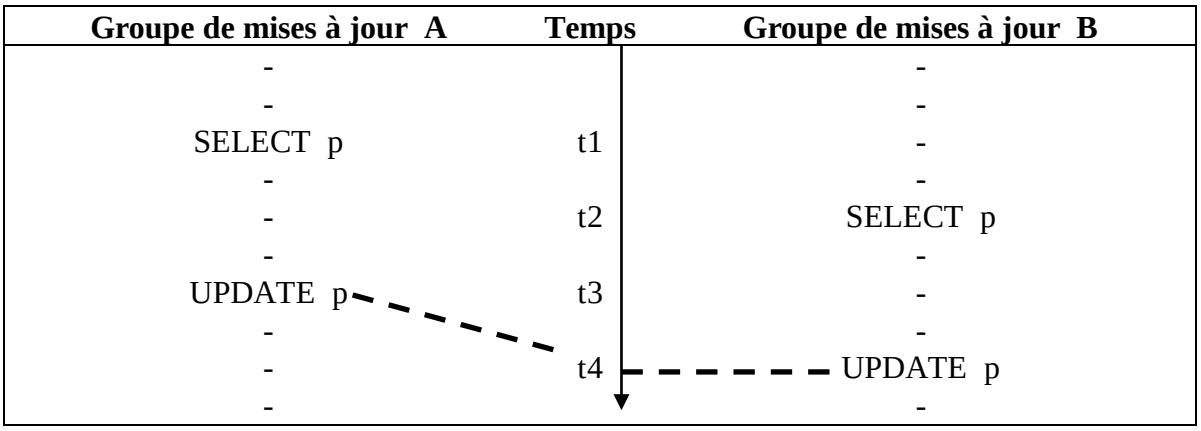
Nous commençons par étudier les différents problèmes que l'on pourrait rencontrer si les transactions n'étaient pas correctement gérées.

Sans réel gestion des transactions, en cas de mises à jour multiples sur les mêmes données : chaque groupe de mises à jour (effectué par 1 utilisateur) pourrait être correct , et toutefois donner un résultat erroné à cause d'interférences de mises à jours exécutées par d'autres utilisateurs.

Ces 3 problèmes sont :

- La perte d'une mise à jour
- Des dépendances non validées
- L'analyse incohérente

① Le problème de la perte d'une mise à jour



A l'instant t4 , la mise à jour du groupe A est perdue, car le groupe B écrit par dessus sans même en avoir pris connaissance.

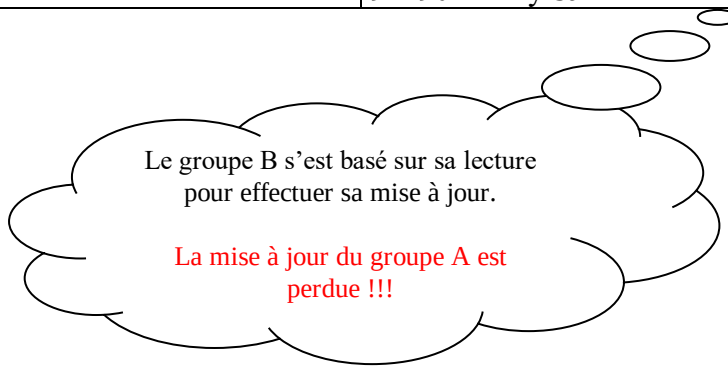
Pourquoi est-ce un problème ?

Supposons que la donnée p soit l'enregistrement correspondant à un client :

Numéro de client	12
Nom	Dupont
Prénom	Jean
Adresse	2, allée des champs fleuris
Code postal	94490
Ville	Vitry-sur-Seine
Taux de remise	1,2

Supposons maintenant les mises à jours suivants correspondants au tableau ci-dessus :

Temps	**GROUPE A**	**GROUPE B**
t1	**lecture de l'enregistrement** 12 'Dupont' 'Jean' '2, allée des champs fleuris' 94490 'Vitry-sur-Seine' 1.2	
t2		**lecture de l'enregistrement** 12 'Dupont' 'Jean' '2, allée des champs fleuris' 94490 'Vitry-sur-Seine' 1.2
t3	**modification du taux de remise qui fait passer le taux de remise à 2** 12 'Dupont' 'Jean' '2, allée des champs fleuris' 94490 'Vitry-sur-Seine' **2**	
t4		**modification de l'adresse du client** 12 'Dupont' 'Jean' **'25, rue Mozart'** 94490 'Vitry-sur-Seine' 1.2



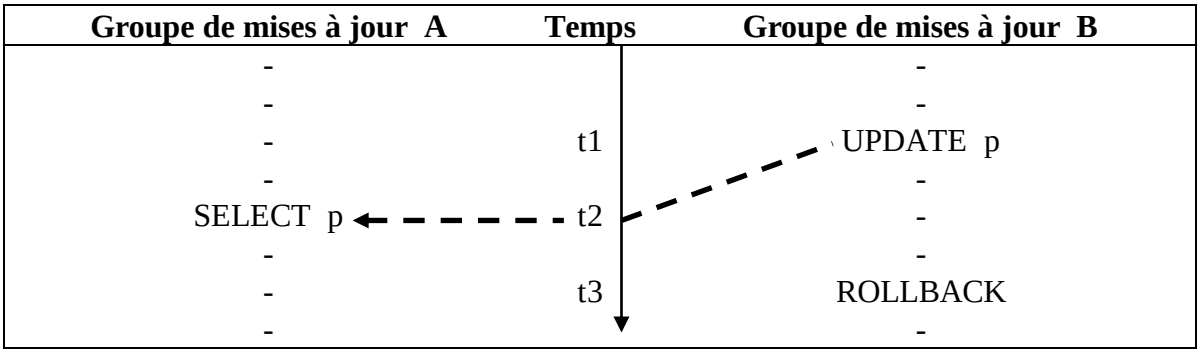
Constat :

Chaque groupe de mises à jour, pris individuellement, est correct.

L'entrelacement des 2 groupes de mises à jour produit un résultat incorrect.

② Le problème des dépendances non validées

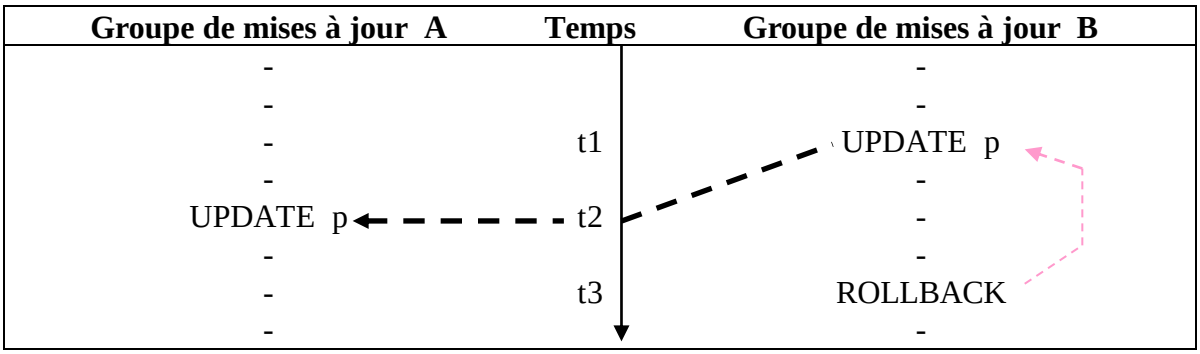
Ce problème apparaît si un groupe de mises à jour lit (ou pire : met à jour) un enregistrement qui vient d'être mise à jour par un autre groupe sans pour autant avoir été validé (COMMIT).



A l'instant t2, le groupe A devient dépendante d'une mise à jour du groupe B, alors que celle-ci n'a pas été validée.

De plus, à l'instant t3 le groupe B annule sa mise à jour. Le groupe A voit une donnée qui n'aurait jamais dû être vue.

Le 2ème exemple est encore pire :



A l'instant t2, le groupe A met à jour une donnée avec des informations non validées.
A l'instant t3, le groupe A perd sa mise à jour, car le groupe B revient à l'état initial de la donnée.

Pourquoi est-ce un problème ?

Basons-nous toujours sur notre enregistrement CLIENT :

12 'Dupont' 'Jean' '2, allée des champs fleuris' 94490 'Vitry-sur-Seine' 1.2

Supposons maintenant les mises à jours suivants correspondants au tableau ci-dessus :

Premier cas

Temps	GROUPE A	GROUPE B
t1		**modification de l'adresse du client** 12 'Dupont' 'Jean' **'25, rue Mozart'** 94490 'Vitry-sur-Seine' 1.2
t2	**lecture de l'enregistrement** 12 'Dupont' 'Jean' **'25, rue Mozart'** 94490 'Vitry-sur-Seine' 1.2	
t3		**ROLLBACK, remet la valeur initiale** 12 'Dupont' 'Jean' '2, allée des champs fleuris' 94490 'Vitry-sur-Seine' 1.2

Le groupe B annule sa mise à jour.

Le groupe A exécute des opérations sur une donnée incorrecte !!!

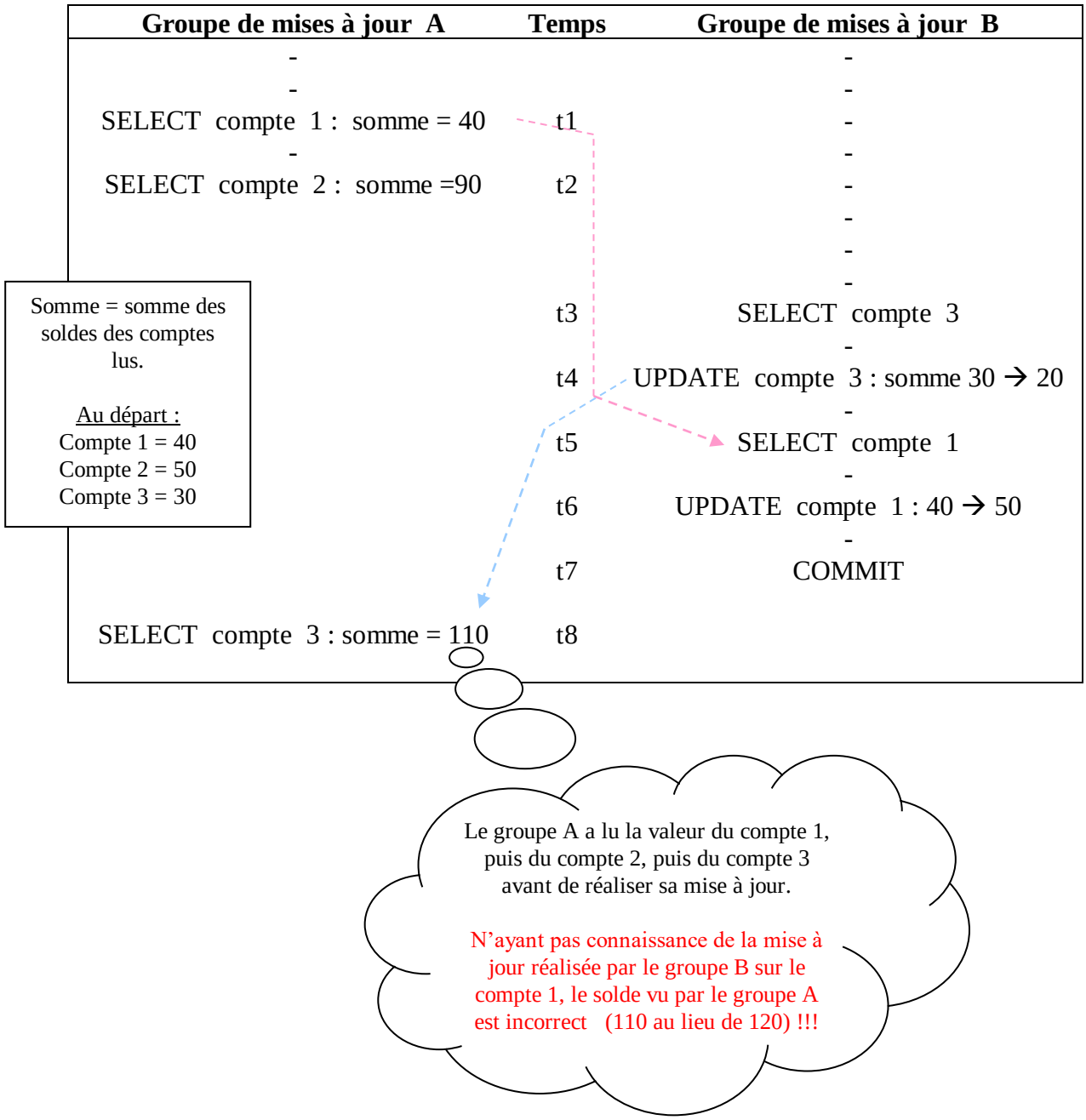
Deuxième cas

Temps	GROUPE A	GROUPE B
t1		**modification de l'adresse du client** 12 'Dupont' 'Jean' **'25, rue Mozart'** 94490 'Vitry-sur-Seine' 1.2
t2	**modification du taux de remise du client** 12 'Dupont' 'Jean' **'25, rue Mozart'** 94490 'Vitry-sur-Seine' **2**	
t3		**ROLLBACK, remet la valeur initiale** 12 'Dupont' 'Jean' '2, allée des champs fleuris' 94490 'Vitry-sur-Seine' 1.2

Le groupe B annule sa mise à jour.

Le groupe A exécute une nouvelle mise à jour sur une donnée incorrecte, puis perd sa mise à jour !!!

③ Le problème de l'analyse incohérente



II. Le verrouillage

Les problèmes présentés dans le premier chapitre peuvent être résolus en utilisant une technique appelée le **verrouillage**

L'idée est simple :

Un groupe de mises à jour (transaction) qui a besoin d'une donnée et qui veut éviter que celle-ci soit modifiée par une transaction **pose un verrou** sur la donnée.

Tant que la transaction qui a posé le verrou ne l'enlève pas, aucune autre transaction pourrait accéder à la donnée 'verrouillée'.

En réalité, il existe différents types de verrous, car parfois on autorise d'autres transactions à accéder à des données verrouillées.

Verrous exclusifs = verrous d'écriture (**X lock**)	Verrous partagés = verrous de lecture (**S lock**)

① Si une transaction A détient un verrou exclusif (X) sur la donnée p, la demande d'un verrou sur la même donnée par la transaction B sera refusée

Donnée	Transaction A	Transaction B
P(-- -- --)	Détient verrou X	Demande verrou (X ou S)

② Si une transaction A détient un verrou partagé (S) sur la donnée p :
- la demande d'un verrou (X) sur la même donnée par la transaction B sera refusée
- la demande d'un verrou (S) sur la même donnée par la transaction B pourra être accordée

Donnée	Transaction A	Transaction B
P(-- -- --)	Détient verrou S	Demande verrou (X) Demande verrou (S)

Introduisons maintenant **le protocole d'accès aux données.**
C'est d'après ce protocole qu'il faut gérer les transactions.

Une transaction qui souhaite **lire une donnée**	Doit obtenir un **verrou S** sur la donnée
Une transaction qui souhaite **mettre à jour une donnée**	Doit obtenir un **verrou X** sur la donnée

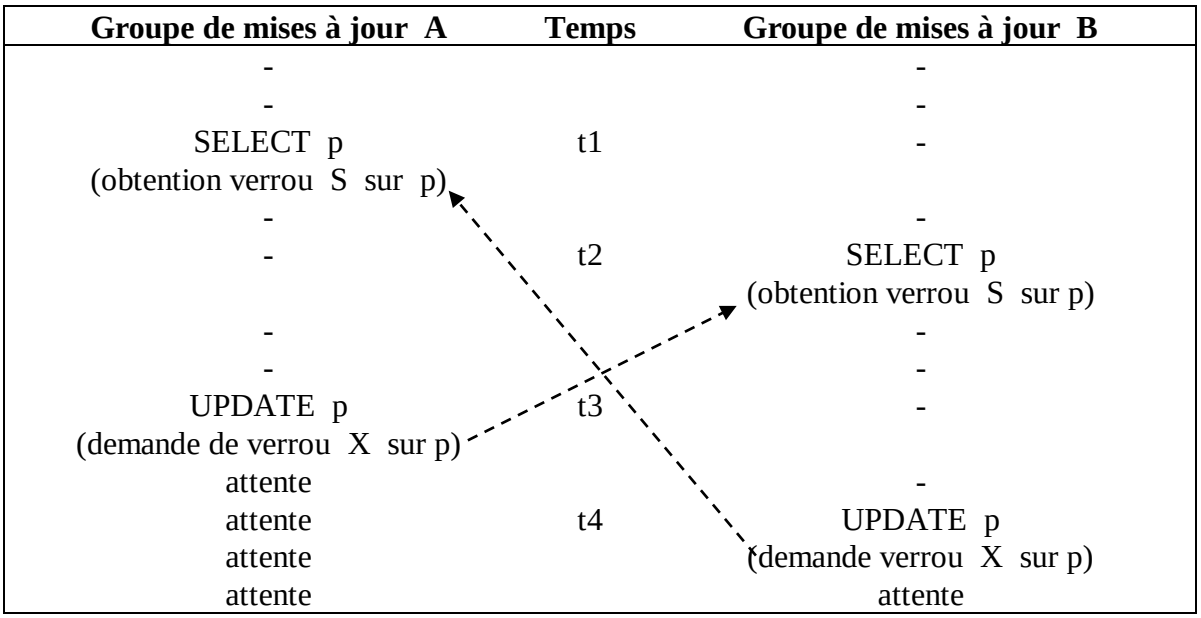
Les demandes de verrous sont en général implicites.
Une consultation d'un enregistrement (SELECT) pose automatiquement un verrou S sur l'enregistrement.
Une mise à jour d'un enregistrement (INSERT, UPDATE, DELETE) pose automatiquement un verrou X sur l'enregistrement.
On peut également poser des verrous de façon explicite dès le début de la transaction.

La demande de verrou d'une transaction est refusée car elle rentre en conflit avec un verrou déjà existant sur la donnée	La transaction est mise en attente jusqu'à libération du verrou déjà existant.

Les verrous X sont conservés jusqu'à la fin de la transaction (COMMIT ou ROLLBACK).

III. Les 3 problèmes de concurrence revisités

① Le problème de la perte d'une mise à jour

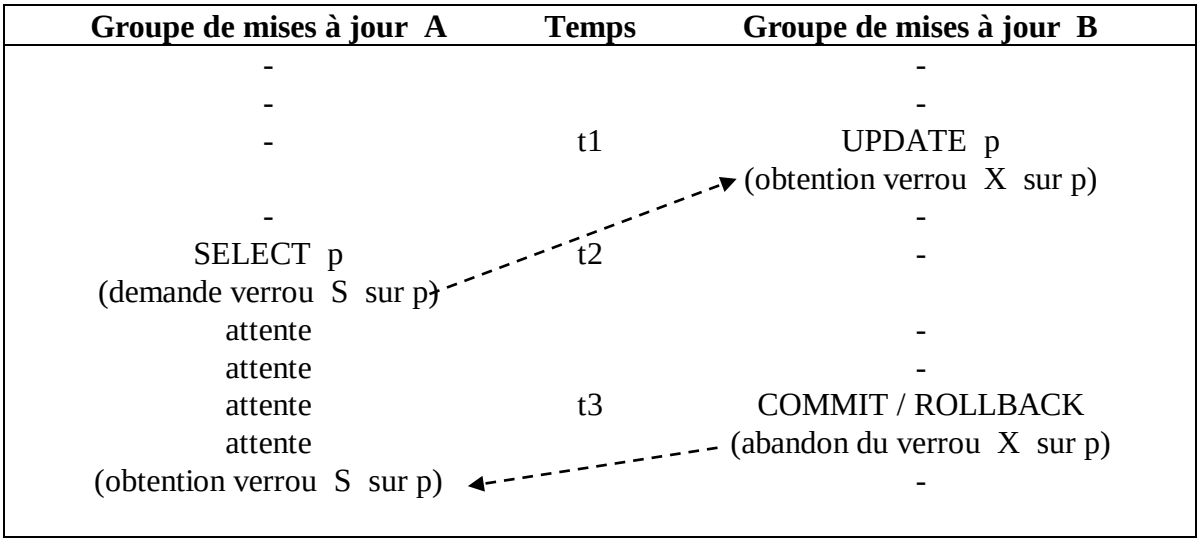


Aucune mise à jour n'est perdue, mais un blocage intervient à l'instant t4.

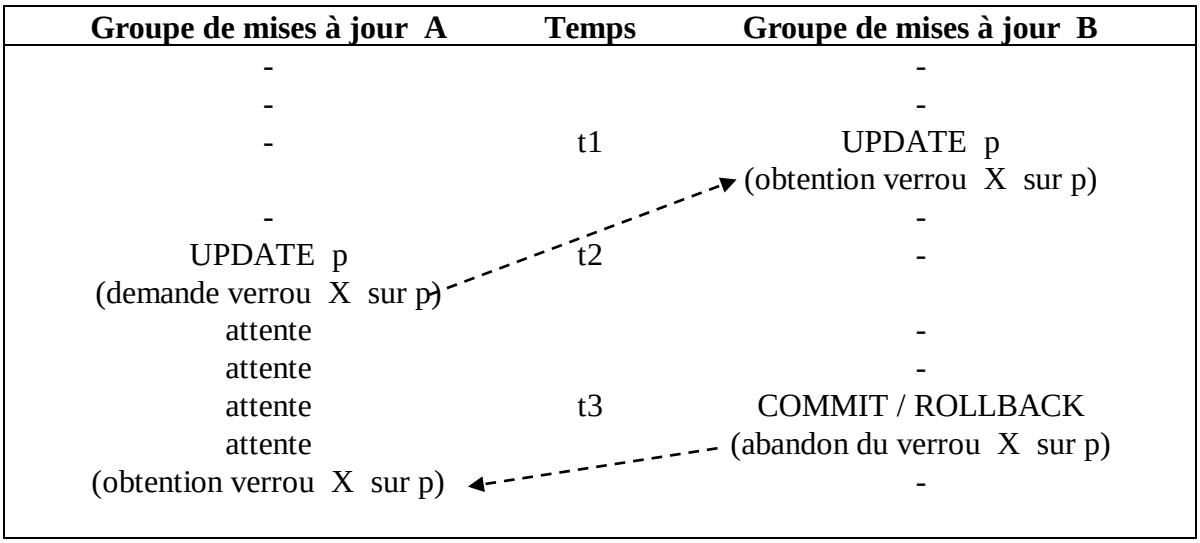
Ce type de blocage est appelé 'étreinte fatale' ou 'deadlock'.

② Le problème des dépendances non validées

Ce problème apparaît si un groupe de mises à jour lit (ou pire : met à jour) un enregistrement qui vient d'être mise à jour par un autre groupe sans pour autant avoir été validé (COMMIT).



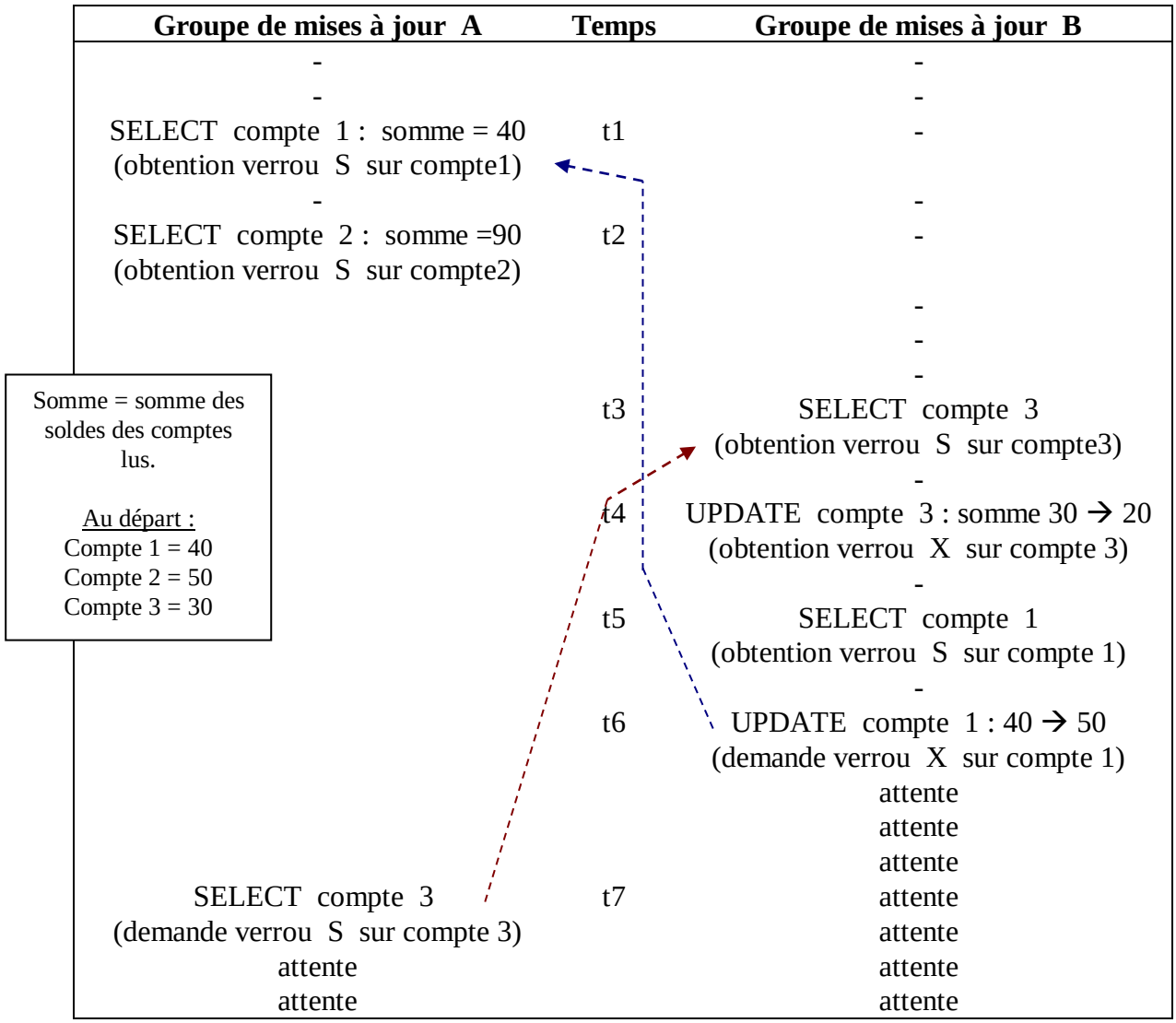
Le 2ème exemple:



Dans les 2 cas :

Lorsque la transaction B abandonne le verrou X sur la donnée p, A peut poursuivre son exécution. A partir de cet instant , la transaction A observera une donnée validée (la valeur initiale en cas de ROLLBACK, la nouvelle valeur en cas de COMMIT).

③ Le problème de l'analyse incohérente



L'analyse incohérente est évitée, mais un blocage survient à l'instant t7.

IV. Le blocage

Le verrouillage peut être utilisé pour résoudre les problèmes de concurrence d'accès aux données. Toutefois, le verrouillage peut, à son tour, apporter un autre problème, celui du blocage.

Le blocage est une situation dans laquelle plusieurs transactions (2 ou plus) sont simultanément en attente. Chaque transaction attend que les autres transactions abandonnent un verrou.

Parmi les SGBDR du marché, plusieurs solutions existent pour régler le problème du blocage.

> Certains détectent le blocage, et analysent qui attend qui depuis le plus de temps. Ils 'tuent' alors la (ou les) transaction(s) qui causent le blocage. A ce moment les verrous posés par ces transactions sont libérés et la transaction restante peut poursuivre son exécution.
>
> D'autres détectent seulement qu'une transaction ne fait rien depuis un certain délai. Partant du point de vu que si une transaction ne fait rien depuis un certain temps, elle doit être bloquée, ces systèmes annulent alors la transaction.

V. La sérialisabilité

La sériabilité est le **critère de correction** généralement accepté pour le contrôle de concurrence.

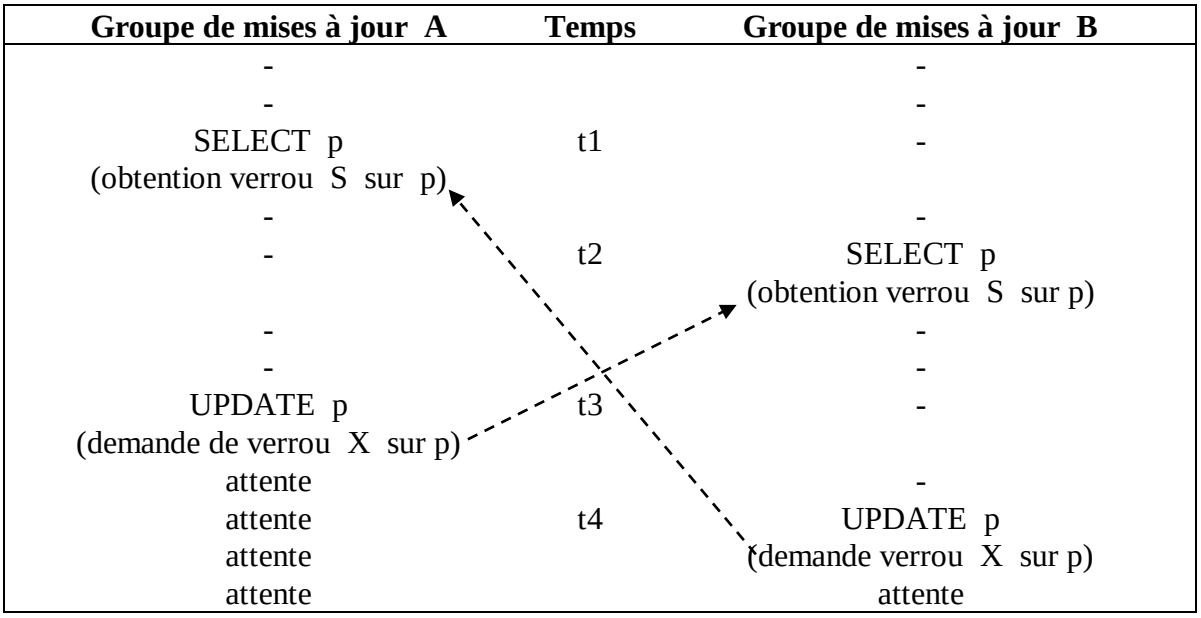
Plus précisément :
Une exécution entrelacée de plusieurs transactions est correcte si elle est sérialisable.
C'est à dire, si elle produit le même résultat qu'une exécution en série des mêmes transactions.

- Prises séparément , les transactions sont supposées correctes. C'est à dire, elles transforment un état cohérent de la base de données en un nouvel état cohérent.
- L'exécution des transactions les unes après les autres, dans n'importe quel ordre séquentiel, est donc également correct ('n'importe quel ordre : les transactions sont supposées être indépendantes les unes des autres).
- Une exécution entrelacée est correcte si elle est équivalente à une exécution en série, c'est à dire si elle est sérialisable.

Le principe de verrouillage est précisément de rendre les transactions sérialisables.

Observons à nouveau les figures utilisés pour expliquer le verrouillage.

Cas N° 1



Dans le cas d'un blocage, une des transactions sera annulé afin de laisser poursuivre l'autre. La transaction annulée devra alors être relancée ultérieurement, c'est à dire après la fin de l'autre transaction. Par conséquent les transactions sont sérialisables.

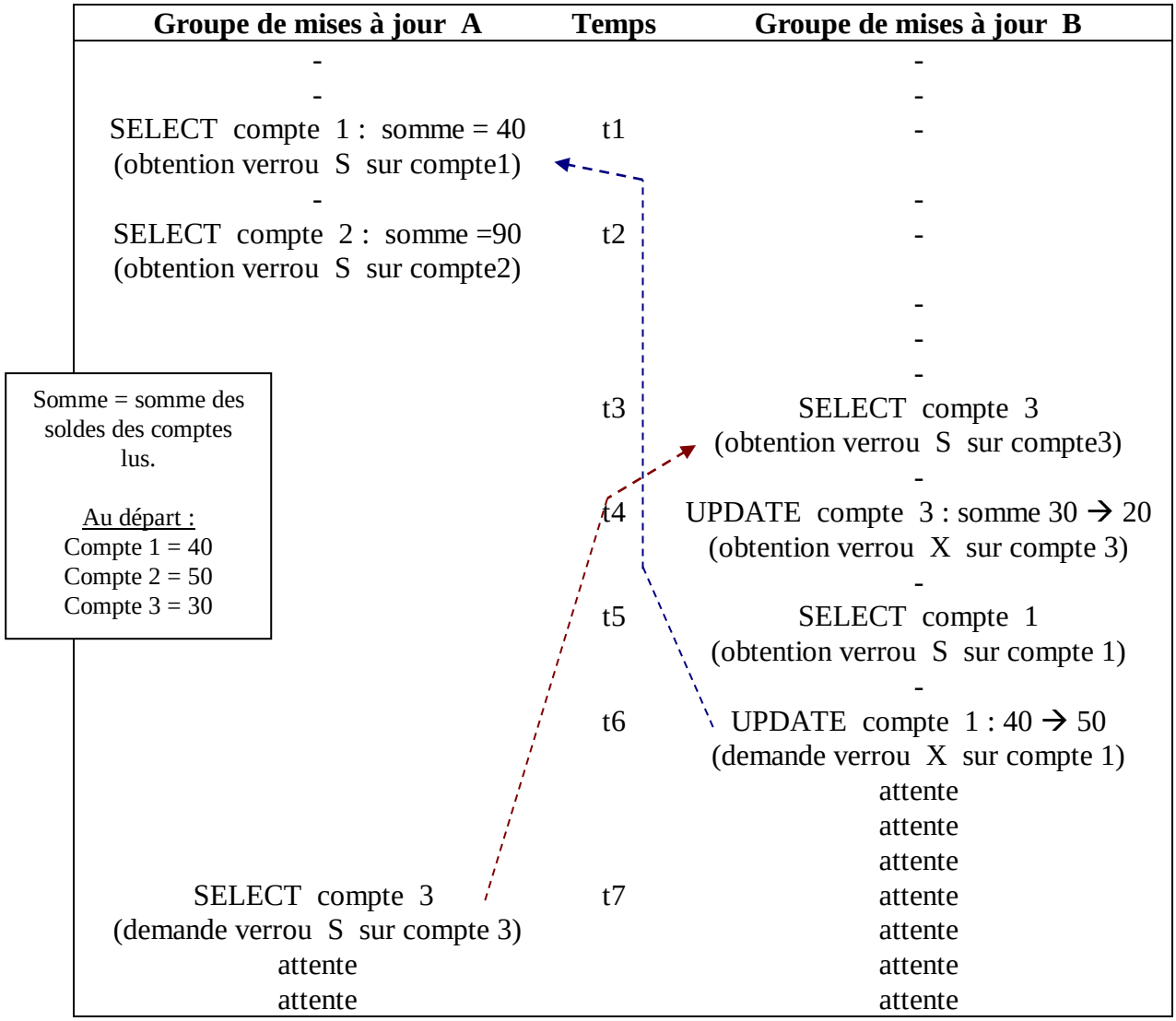
Cas N° 2

Groupe de mises à jour A	Temps	Groupe de mises à jour B
-		-
-		-
-	t1	UPDATE p
		(obtention verrou X sur p)
-		-
SELECT p	t2	-
(demande verrou S sur p)		
attente		-
attente		-
attente	t3	COMMIT / ROLLBACK
attente		(abandon du verrou X sur p)
(obtention verrou S sur p)		-

Groupe de mises à jour A	Temps	Groupe de mises à jour B
-		-
-		-
-	t1	UPDATE p
		(obtention verrou X sur p)
-		-
UPDATE p	t2	-
(demande verrou X sur p)		
attente		-
attente		-
attente	t3	COMMIT / ROLLBACK
attente		(abandon du verrou X sur p)
(obtention verrou X sur p)		-

La transaction A attend la fin de la transaction B.
Par conséquent, cela est équivalent à une exécution en série : B suivi de A.
Les transactions sont donc sérialisables.

Cas N° 3



Dans le cas d'un blocage, une des transactions sera annulé afin de laisser poursuivre l'autre. La transaction annulée devra alors être relancée ultérieurement, c'est à dire après la fin de l'autre transaction. Par conséquent les transactions sont sérialisables.

VI. Le niveau d'isolation

Le terme **niveau d'isolation** est utilisé pour désigner le **degré d'interférence** qu'une transaction est prête à tolérer avec les autres transactions concurrentes.

Si l'on veut garantir la sérialisabilité, on ne doit pas accepter d'interférence !

Cependant, pour des raisons pragmatiques diverses, les SGBDR du marché autorisent des transactions acceptant des interférences à un certain degré.