TD: Transaction / Concurrence et pannes

Exercice 1. **Atomicité / Durabilité**
Soit un programme de transfert entre comptes bancaires, correspondant au schéma suivant :

procédure *transfert* (CompteDébité, CompteCrédité, montant)
*(1) variable := **Lire**(CompteDébité);*
*(2) **Ecrire** (CompteDébité, variable – montant);*
*(3) variable := **Lire** (CompteCrédité);*
*(4) **Ecrire**(CompteCrédité, variable + montant)*
fin ;

1.1. Ecrire en SQL la série d'instructions (1) à (4), dans le cas d'une base de données comportant une relation COMPTE, de schéma COMPTE (NumCompte, NumClient, Solde).

1.2.. Soient deux transactions, T0 et T1, exécutant l'une après l'autre *transfert* (A,B,100) et *transfert*(B,C,200), respectivement. Les valeurs initiales des comptes A, B, et C étant respectivement 1000, 500, et 300, quel est le contenu du journal et l'état de la base lorsque les deux transactions terminent leur exécution avec succès dans le cas d'une écriture en mode différé ?

1.3. Toujours dans le cas d'une écriture en mode différé, supposons maintenant qu'une panne système survient à trois moments différents:
(a) juste après l'écriture dans le journal de l'enregistrement correspondant à la 4ème instruction de T0,
(b) juste après l'écriture de l'enregistrement correspondant à la 4ème instruction de T1,
(c) juste après l'écriture de l'enregistrement <T1, commit> (i.e., après l'exécution complète et la validation des deux transactions).
Sachant que les écritures dans la base sont faites en mémoire cache, laquelle n'est systématiquement recopiée sur disque qu'en certaines occasions, montrer pour chacun des cas de panne (a), (b), et (c), le contenu du journal et expliquer les opérations à effectuer pour assurer la cohérence de la base après la reprise du système.
1.4. Même question que 1.2. dans le cas d'une écriture en mode immédiat.
1.5. Même question que 1.3. dans le cas d'une écriture en mode immédiat.

Exercice 2 : **Isolation et sérialisabilité**
2.1. Soit transfert la procédure de transfert entre comptes bancaires introduite dans l'exercice 1.
On lance simultanément les programmes *transfert*(A, B, 150) et transfert(B, C, 70). Chacun des processus (i.e., chacune des transactions de transfert) utilise une variable locale à son propre espace mémoire. Soient A0, B0, C0 les états initiaux des trois comptes concernés. Quel est l'état final des comptes A, B et C dans chacun des cas ci-dessous, où ABi représente l'ième instruction du transfert de A vers B et BCj la jème instruction du transfert de B vers C?

cas (a) = exécution complète de *transfert*(B,C,70), puis exécution de *transfert*(A,B,150)
cas (b) = exécution des instructions dans l'ordre
AB1 BC1 BC2 BC3 AB2 AB3 AB4 BC4
cas (c) = exécution des instructions dans l'ordre
AB1 AB2 AB3 BC1 BC2 BC3 AB4 BC4

2.2. Quelle contrainte d'intégrité n'est pas respectée par l'une des exécutions précédentes ?

2.3. Dans chacun des deux derniers cas, peut-on obtenir une exécution séquentielle des deux programmes de transfert, par permutation d'opérations successives non conflictuelles ?

2.4. Donner pour chacun des cas (a), (b) et (c) le graphe de précédence du transfert de A vers B, et du transfert de B vers C.

Exercice 3. Verrouillage en deux phases
Soit la procédure transfert de l'exercice 1.2 dans laquelle on a inséré des instructions de verrouillage et de déverrouillage selon le protocole de verrouillage en deux phases strict.

3.1. Ecrire la série d'instructions obtenue, en supposant que deux comptes différents appartiennent à des granules différents.

3.2. Les exécutions (b) et (c), décrites dans l'exercice précédent, sont-elles alors possibles ? Si non, donner une exécution possible ayant le même ordonnancement initial jusqu'à la première instruction non réalisable.