TD 06 : Transactions

Exercice 1

Soit un ensemble de comptes bancaires et soit les procédures suivantes.

- La procédure *Depot* réalisant un dépôt sur un compte bancaire :

```
procedure Dépôt(num, Montant)
begin
 temp := Read(Comptes(num)) ;
 temp := temp + Montant ;
 Write(Comptes(num),temp) ;
end ;
```

- La procédure *ImpSomme* réalisant la somme des soldes sur deux comptes passés en paramètre :

```
procedure ImpSomme(Compte1,Compte2)
begin
   temp1 := Read(Comptes(Compte1)) ;
   print(temp1) ;
   temp2 := Read(Comptes(Compte2)) ;
   print(temp2) ;
   temp1 := temp1 + temp2 ;
   print(temp1) ;
end ;
```

a) Au départ le compte `13` a un solde de `10.000€`. Supposons que le client `1` dépose `1.000€` sur le compte `13` et que le client `2` y dépose `1500€`. Chaque client appelle la procédure *`Depot`* et crée une transaction (respectivement T_1 et T_2). L'exécution simultanée de T_1 et T_2 se passe comme suit (on n'indique que les opérations communiquant avec la base de données) :

Exécution 1

```
T1: Read(comptes(13)) ;
T2: Read(comptes(13)) ;
T2: Write(comptes(13), 11500 );
T1: Write(comptes(13), 11000);
```

Question 01 : Quel est le résultat de cette exécution ? Que remarquez-vous ?

b) Supposons maintenant que les comptes `7` et `8` ont des soldes de `20000` euros et que le client `1` affiche les soldes des deux comptes en utilisant la procédure *`ImpSomme`* (transaction T_1). En même temps, le client `2` transfère `1000` euros du compte `7` vers le compte `8` (transaction T_2) en utilisant la procédure *`Transfert`*

suivante :

```
procedure Transfert(Compte1, Compte2, Montant)
begin
   temp := Read(Comptes(Compte1)) ;
   temp := temp – Montant ;
   Write(Comptes(Compte1),temp) ;
   temp := Read(Comptes(Compte2)) ;
   temp := temp + Montant ;
   Write(Comptes(Compte2),temp) ;
end ;
```

L'ordonnancement des opérations est donné par l'exécution suivante :

Exécution 2

(1) T_2: Read(Comptes(7)) ;
(2) T_2: Write(Comptes(7),19000) ;
(3) T_1: Read(Comptes(7)) ;
(4) T_1: Read (Comptes(8)) ;
(5) T_2: Read(Comptes(8)) ;
(6) T_2: Write(Comptes(8),21000).

Question 02 :Quel est le résultat de cette exécution ? Que pouvez-vous dire sur les valeurs affichées à l'écran ?

Question 03 : Mêmes questions pour l'exécution suivante.

Exécution 3

(1) T_2: Read (Comptes(7)) ;
(2) T_2: Write(Comptes(7),19000) ;
(3) T_1: Read (Comptes(7)) ;
(4) T_2: Read (Comptes(8)) ;
(5) T_2: Write(Comptes(8),21000) ;
(6) T_1: Read (Comptes(8)) ;

Exercice 2

Une granule est une unité de donnée dont l'accès est contrôlée individuellement par le SGBD. Soient T_1, T_2, T_3, T_4, T_5 et T_6 six transactions et x, y, z et t quatre granules d'une base de données.

a) Construire le graphe de précédence de l'exécution suivante, en indiquant sur chaque flèche du graphe le ou les granules correspondant à la précédence. Est-elle sérialisable ? Si oui, donner toutes les exécutions en série équivalente(s).
(1) T_1: Write(t),
(2) T_2: Read(z),

(3) T_4: Read(z),
(4) T_2: Write(y),
(5) T_4: Write(z),
(6) T_2: Read(t),
(7) T_2: Read(x),
(8) T_3: Write(x).

b) Construire le graphe de précédence de l'exécution suivante. Est-elle sérialisable ? Si oui, donner la ou les exécutions en série équivalente(s).
(1) T_1: Read(z),
(2) T_4: Read(z),
(3) T_1: Write(y),
(4) T_4: Write(z),
(5) T_1:Write(x),
(6) T_2: Read(y),
(7) T_3: Read (y),
(8) T_4: Write(y),
(9) T_2: Read(z),
(10) T_2: Read(x),
(11) T_3: Write(z).

c) Construire le graphe de précédence de l'exécution suivante. Est-elle sérialisable ? Si oui, donner la ou les exécutions en série équivalente(s).
(1) T_1: Read (x),
(2) T_1: Read (y),
(3) T_2: Read (x),
(4) T_2: Write(z),
(5) T_3: Write(x),
(6) T_4: Write(y),
(7) T_4: Write(x),
(8) T_5: Read (y)
(9) T_3: Read (z),
(10) T_6: Read (y),
(11) T_4: Read (z).

Exercice 3

Soit 3 transactions T_1, T_2 et T_3, 3 granules X, Y et Z. L correspond à l'opération de lecture et E à l'opération d'écriture. Soit les 4 exécutions suivantes H_1, H_2, H_3 et H_4 :

- H_1: $L_1(x)$, $L_2(x)$, $E_3(z)$, $E_2(y)$, $E_1(x)$, $E_1(z)$, $L_3(y)$
- H_2: $L_1(x)$, $E_2(y)$, $L_3(y)$, $E_3(z)$, $E_1(z)$, $E_1(x)$, $L_2(x)$
- H_3: $E_3(z)$, $E_1(z)$, $L_1(x)$, $E_2(y)$, $L_2(x)$, $L_3(y)$, $E_1(x)$
- H_4: $L_2(x)$, $E_3(z)$, $E_2(y)$, $L_1(x)$, $L_3(y)$, $E_1(x)$, $E_1(z)$

a) Construire le graphe de précédence des exécutions H_1, H_2, H_3 et H_4.
b) Quelles sont les exécutions sérialisables ?
c) Certaines de ces exécutions sont-elles équivalentes ?

Exercice 4 : Transactions Oracles

On considère une table T(A integer, B integer) contenant avant le début des transactions les n-uplets `(1,1)` et `(2,2)`. Tous les utilisateurs ont le droit d'interroger ou de modifier la table T. On considère deux clients p_1 et p_2 qui effectuent les séquences d'ordres suivantes, immédiatement après leur connexion. Un autre client quelconque est représenté par p_3. On considère que niveau d'isolation des transactions est celui par défaut (sous ORACLE: `READ COMMITED`).

Remplir les tables en fonctions des requêtes exécutées. Indiquer les données telles qu'elles sont vues par les utilisateurs p_1, p_2 et p_3.

```
i.
   1  (p1) : delete t where b=1
   2  (p1) : commit
ii.
   1  (p1) : delete t where b=1
   2  (p2) : update t set b=b-1
   3  (p1) : commit
   4  (p2) : commit
iii.
   1  (p2) : update t set b=b-1
   2  (p1) : delete t where b=1
   3  (p2) : commit
   4  (p1) : commit
iv.
   1  (p1) : update t set a = 1 where b = 2
   2  (p2) : delete t where a = 1
   3  (p1) : commit
   4  (p2) : commit
v.
   1  (p1) : update t set a = a-1 where a = 1
   2  (p2) : update t set a = a+1 where a = 2
   3  (p1) : update t set a = a-1 where a = 2
   4  (p2) : update t set a = a+1 where a = 1
   5  (p1) : commit
   6  (p2) : commit
```